Speeding Up Linux Disk Encryption

Ignat Korchagin
@ignatkn

$ whoami

- Performance and security at Cloudflare
- Passionate about security and crypto
- Enjoy low level programming

Encrypting data at rest

The storage stack

applications

The storage stack

applications

filesystems

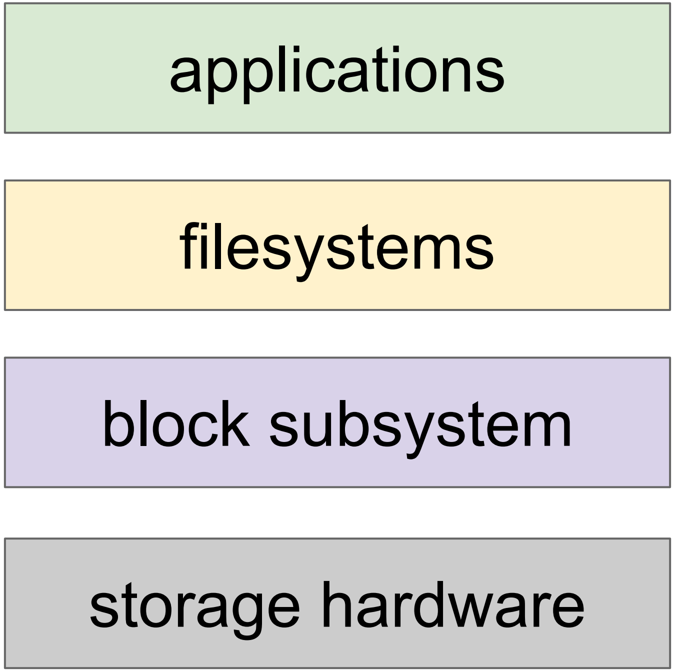
The storage stack

applications

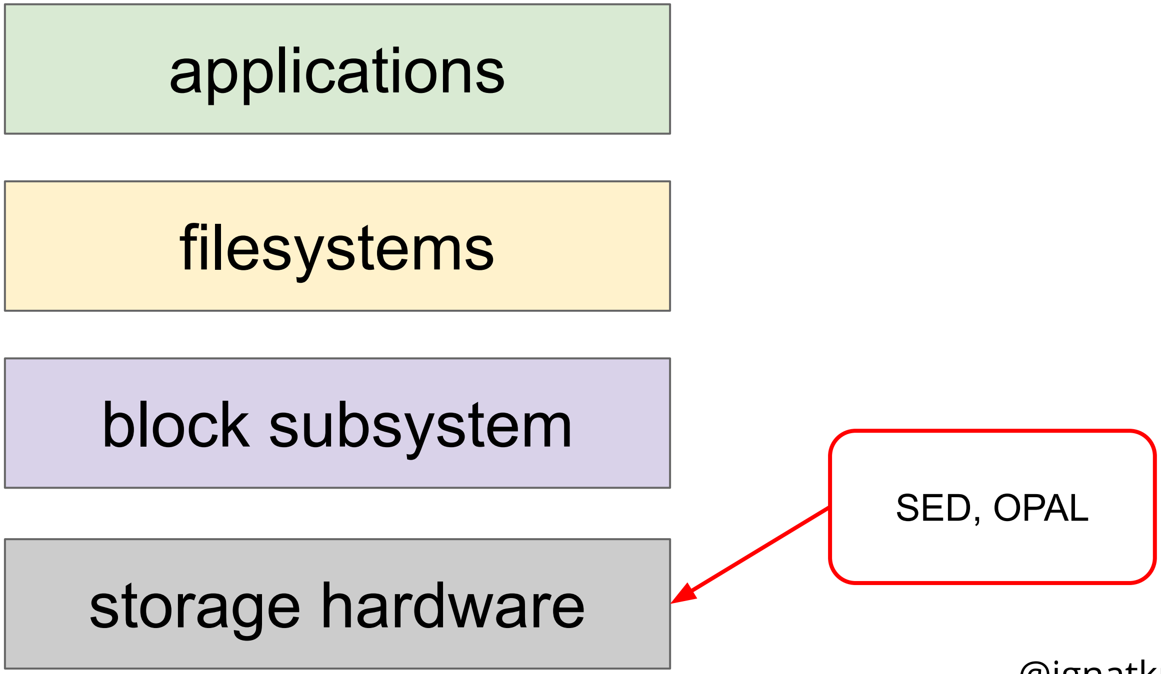
filesystems

block subsystem

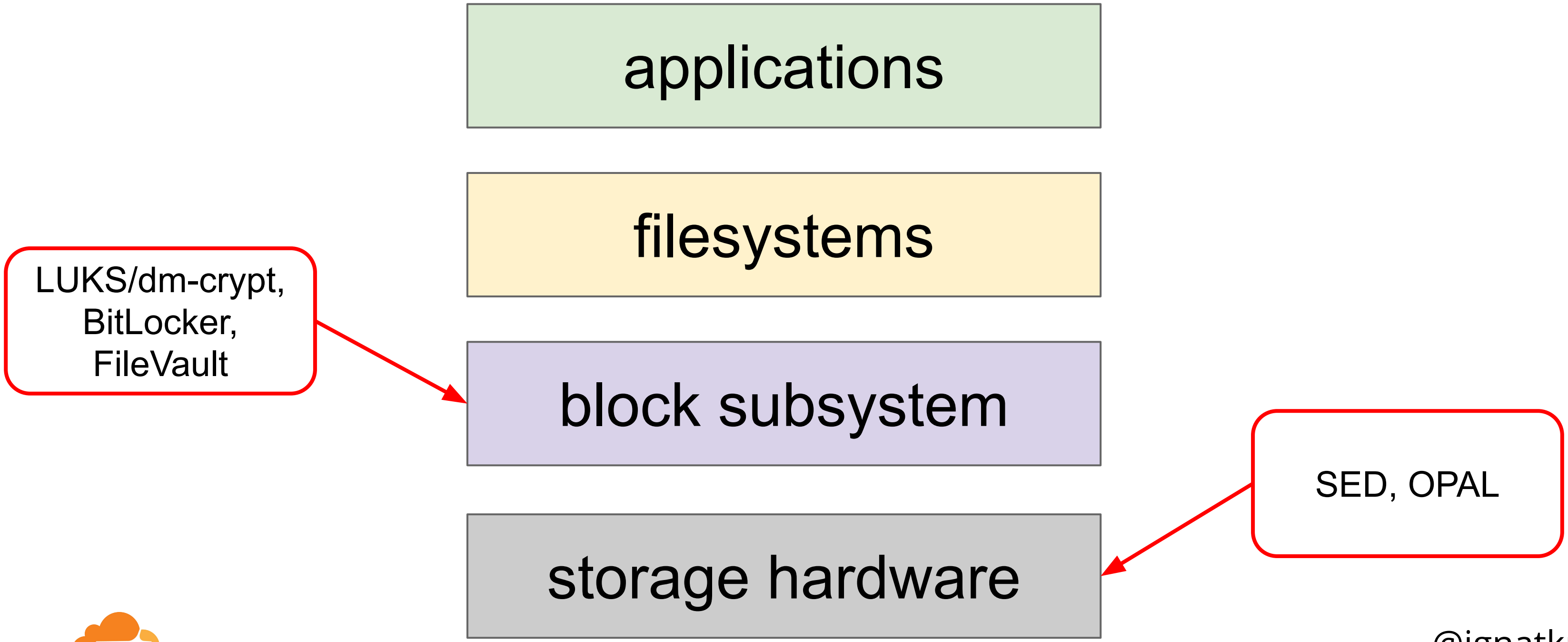
The storage stack



Encryption at rest layers

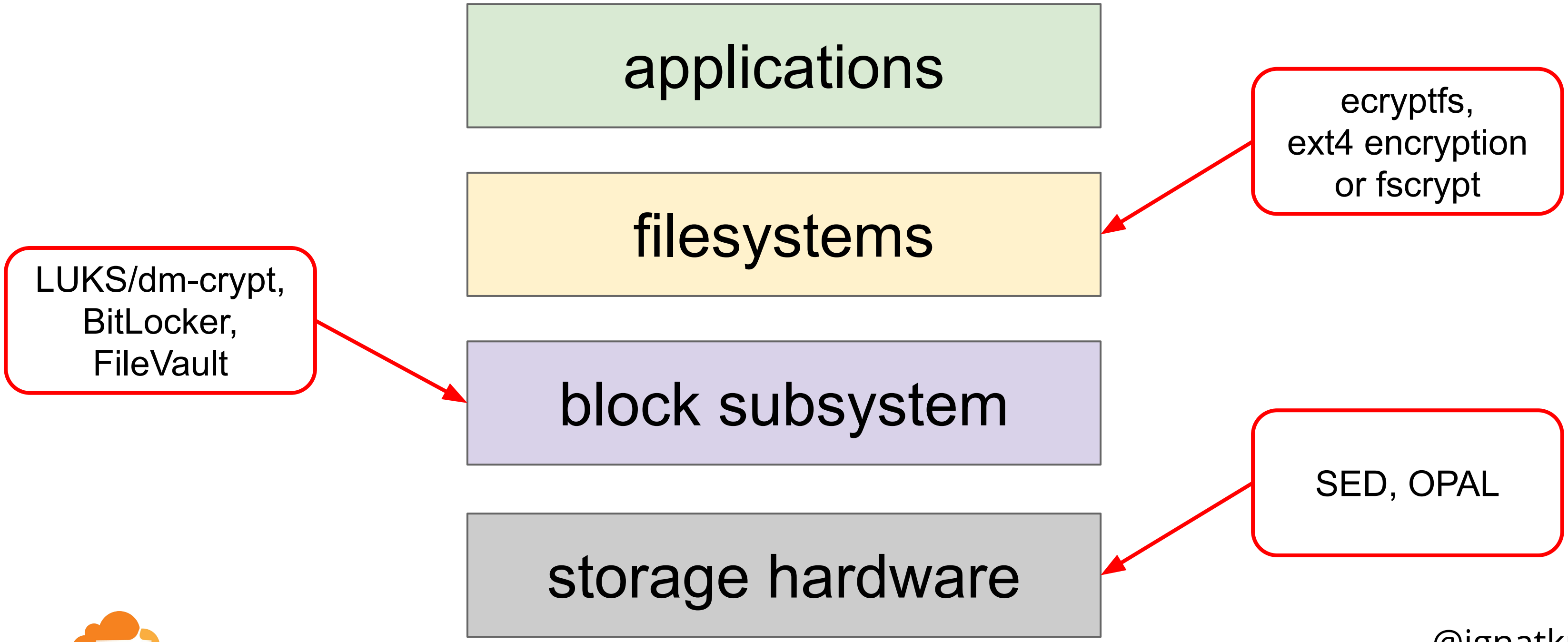


Encryption at rest layers



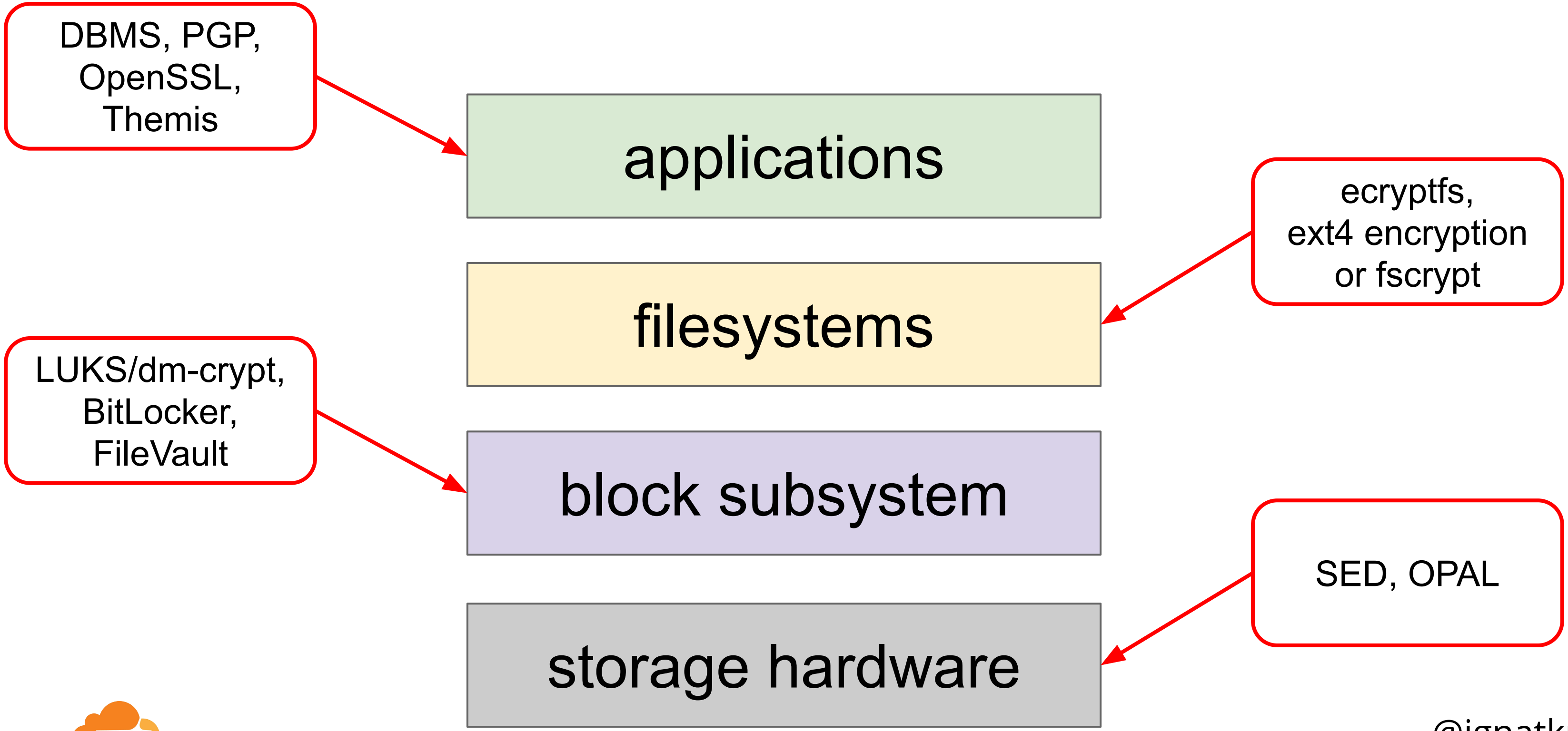
@ignatkn

Encryption at rest layers



@ignatkn

Encryption at rest layers



Storage hardware encryption

Pros:

- it’s there
- little configuration needed
- fully transparent to applications
- usually faster than other layers

Storage hardware encryption

Pros:

- it's there
- little configuration needed
- fully transparent to applications
- usually faster than other layers

Cons:

- no visibility into the implementation
- no auditability
- sometimes poor security

https://support.microsoft.com/en-us/help/4516071/windows-10-update-kb4516071

Block layer encryption

Pros:

- little configuration needed
- fully transparent to applications
- open, auditable

Block layer encryption

Pros:

- little configuration needed
- fully transparent to applications
- open, auditable

Cons:

- requires somewhat specialised crypto
- performance issues
- encryption keys in RAM

Filesystem layer encryption

Pros:

- somewhat transparent to applications
- open, auditable
- more fine-grained
- more choice of crypto + potential integrity support

Filesystem layer encryption

Pros:

- somewhat transparent to applications
- open, auditable
- more fine-grained
- more choice of crypto + potential integrity support

Cons:

- performance issues
- encryption keys in RAM
- complex configuration
- unencrypted metadata

Application layer encryption

Pros:

- open, auditable
- fine-grained
- full crypto flexibility

Application layer encryption

Pros:

- open, auditable
- fine-grained
- full crypto flexibility

Cons:

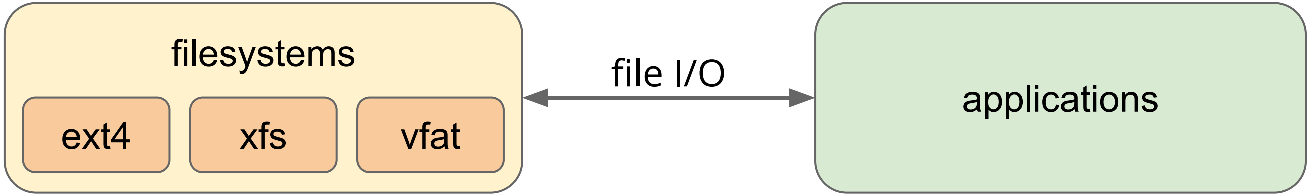
- encryption keys in RAM
- requires explicit support in code and configuration
- unencrypted metadata
- full crypto flexibility

LUKS/dm-crypt

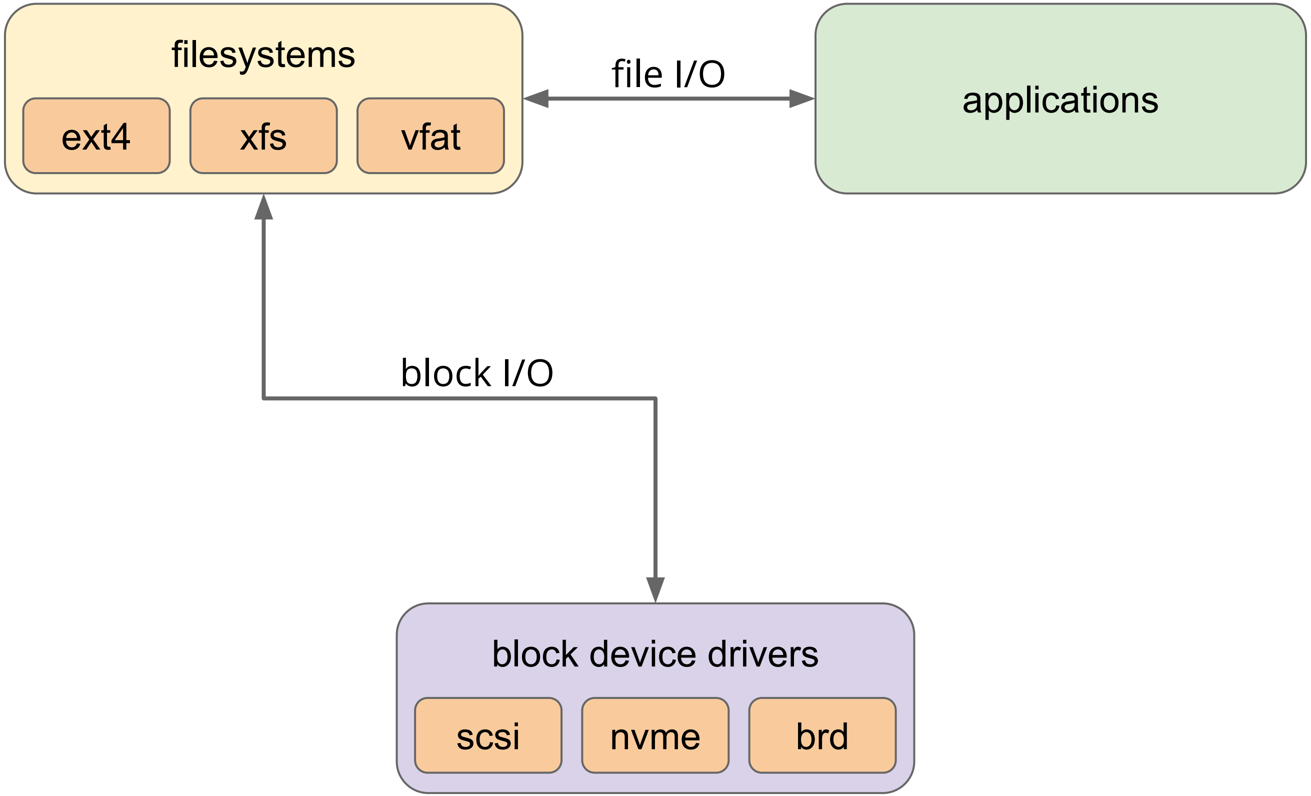
Device mapper in Linux

applications

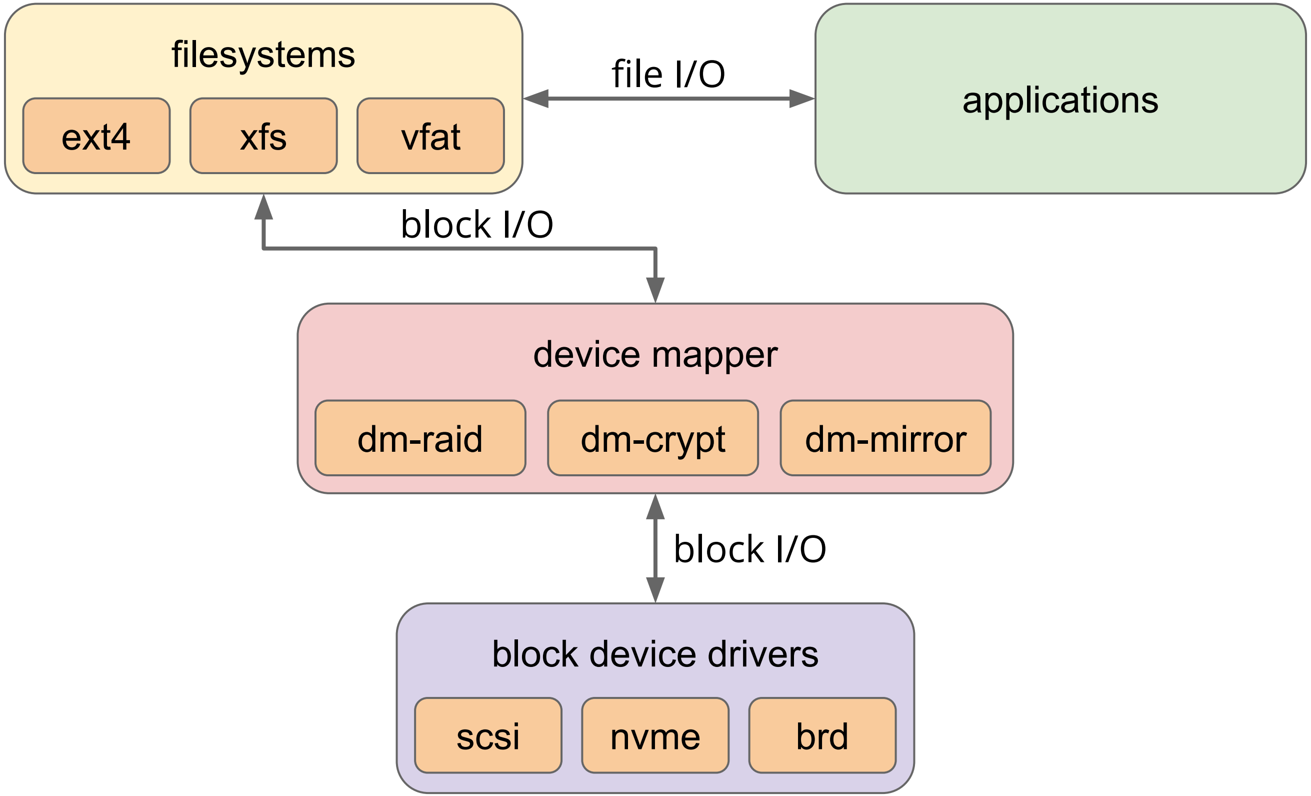
Device mapper in Linux



Device mapper in Linux



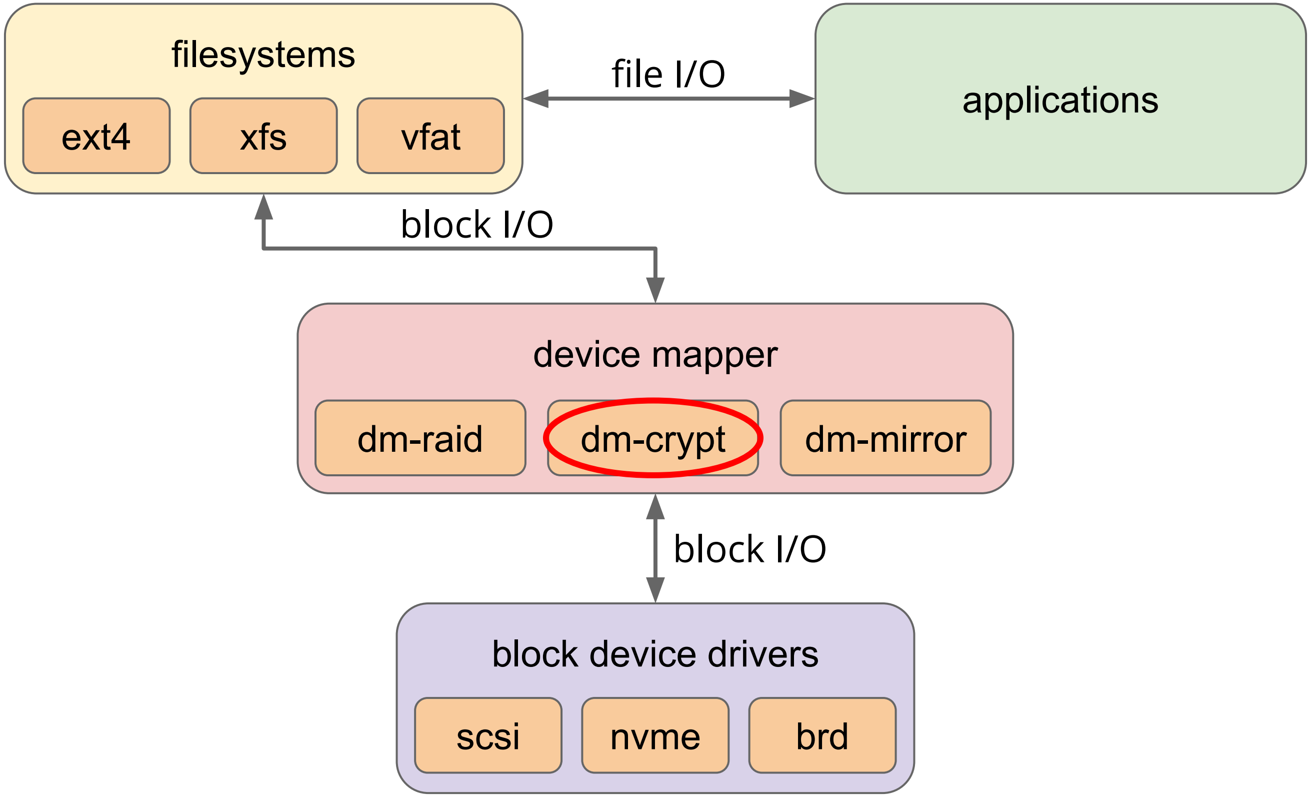
Device mapper in Linux



CLOUDFLARE

@ignatkn

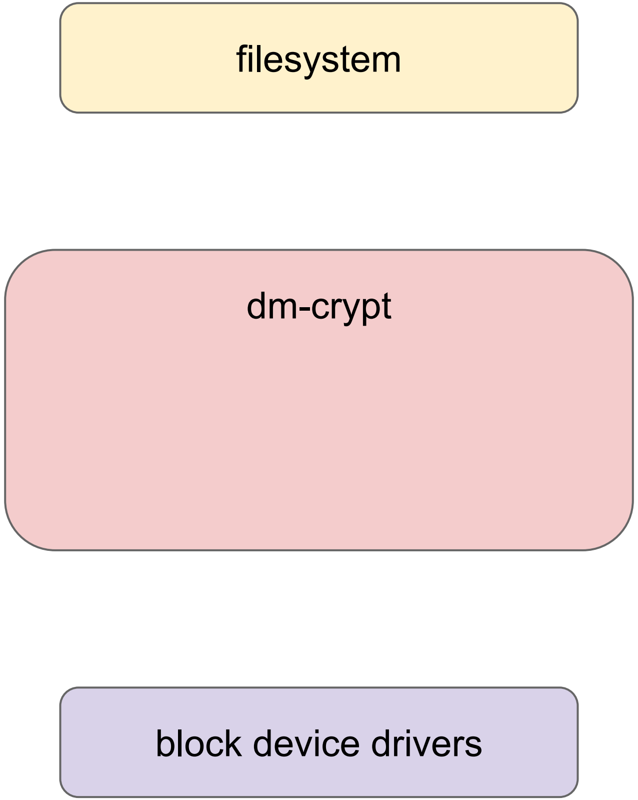
Device mapper in Linux



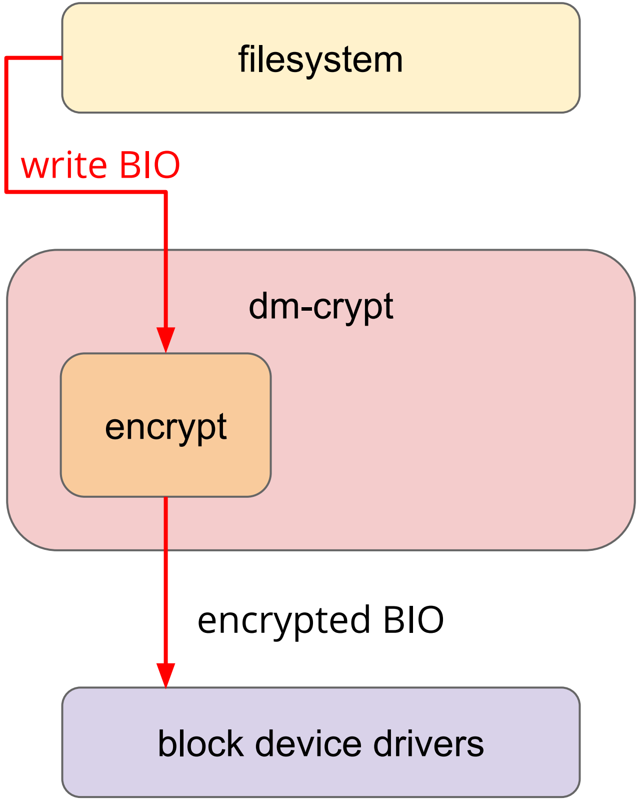
@ignatkn

dm-crypt (idealized)

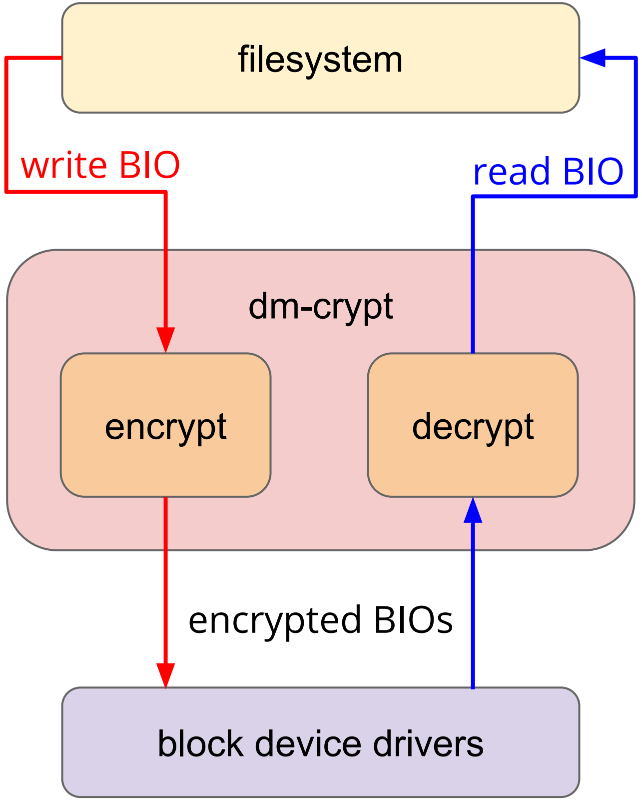
dm-crypt (idealized)



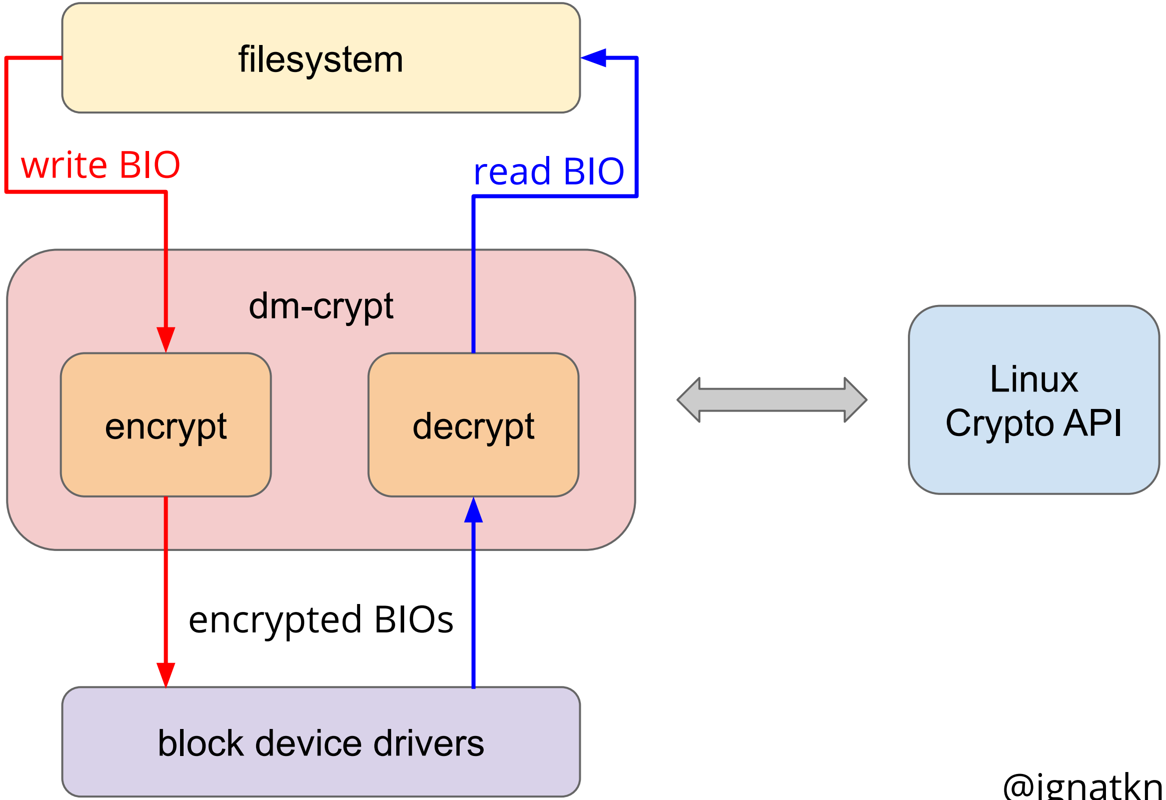
dm-crypt (idealized)



dm-crypt (idealized)



dm-crypt (idealized)



@ignatkn

dm-crypt benchmarking

Test setup: RAM-based encrypted disk

```
$ sudo modprobe brd rd_nr=1 rd_size=4194304
```

Test setup: RAM-based encrypted disk

```
$ sudo modprobe brd rd_nr=1 rd_size=4194304
$ echo '0 8388608 delay /dev/ram0 0 0' |
sudo dmsetup create plain
```

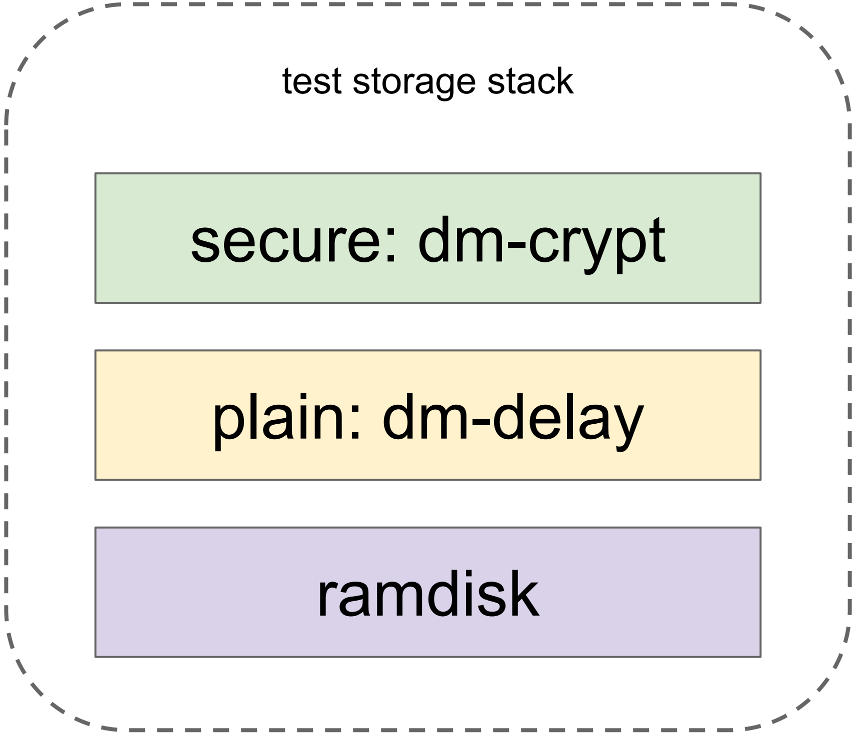
Test setup: RAM-based encrypted disk

```
$ sudo modprobe brd rd_nr=1 rd_size=4194304
$ echo '0 8388608 delay /dev/ram0 0 0' |
sudo dmsetup create plain
$ sudo cryptsetup luksFormat
/dev/mapper/plain
```

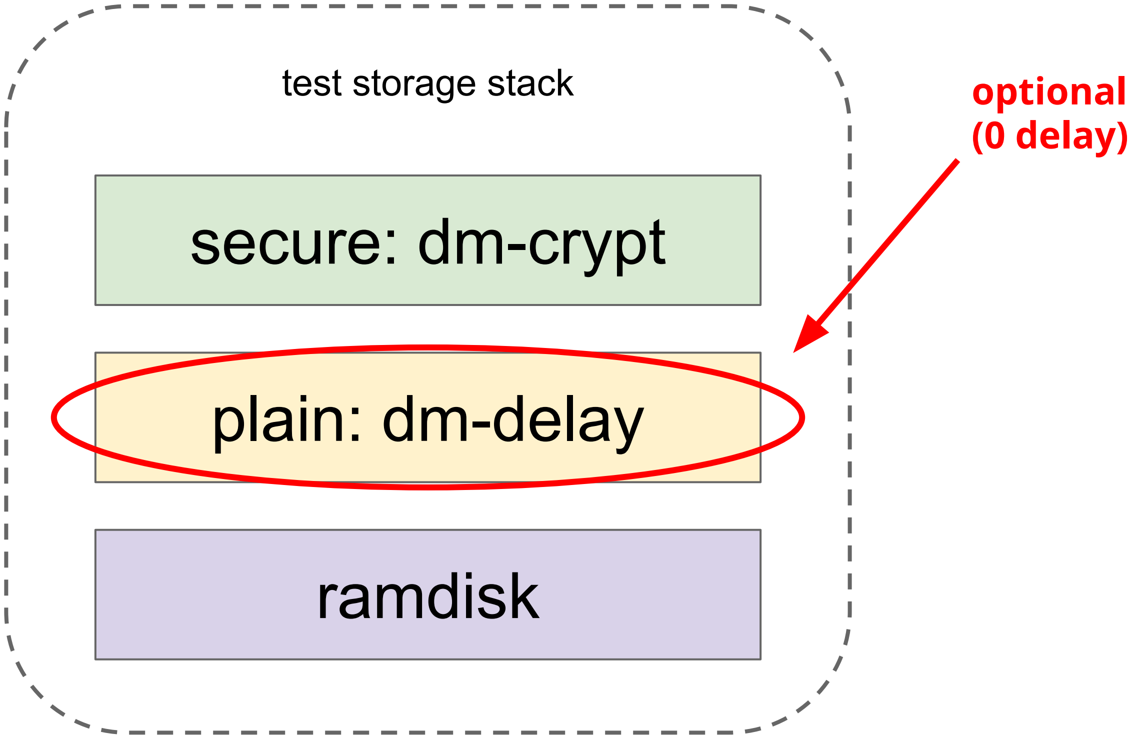
Test setup: RAM-based encrypted disk

```
$ sudo modprobe brd rd_nr=1 rd_size=4194304
$ echo '0 8388608 delay /dev/ram0 0 0' |
sudo dmsetup create plain
$ sudo cryptsetup luksFormat
/dev/mapper/plain
$ sudo cryptsetup open --type luks
/dev/mapper/plain secure
```

Test storage stack



Test storage stack



Test setup: sequential reads

```
$ cat rw.job
[iotest]
direct=1
gtod_reduce=1
loops=1000000
iodepth=16
```

Test setup: sequential reads

```
$ sudo fio --filename=/dev/mapper/plain
--readwrite=read --bs=4k rw.job
...
   READ: io=21134MB, aggrb=1876.1MB/s
```

Test setup: sequential reads

```
$ sudo fio --filename=/dev/mapper/plain
--readwrite=read --bs=4k rw.job

...
   READ: io=21134MB, aggrb=1876.1MB/s
$ sudo fio --filename=/dev/mapper/secure
--readwrite=read --bs=4k rw.job

...
   READ: io=3261.8MB, aggrb=318.6MB/s
```

CLOUDFLARE

@ignatkn

Test setup: sequential reads

```
$ sudo cryptsetup benchmark -c aes-xts
# Tests are approximate using memory only (no
storage IO).
#  Algorithm | Key |  Encryption |  Decryption
     aes-xts   256b  1854.7 MiB/s  1904.5 MiB/s
```

Test setup: sequential reads

```
$ sudo cryptsetup benchmark -c aes-xts
# Tests are approximate using memory only (no
storage IO).
#  Algorithm | Key |   Encryption |   Decryption
      aes-xts   256b  1854.7 MiB/s  1904.5 MiB/s
```

desired: **~900 MB/s**, actual: **~300 MB/s**

We tried...

- switching to different cryptographic algorithms
  - aes-xts seems to be the fastest (at least on x86)

We tried...

- switching to different cryptographic algorithms
  - aes-xts seems to be the fastest (at least on x86)
- experimenting with dm-crypt optional flags
  - “`same_cpu_crypt`” and “`submit_from_crypt_cpus`”

We tried...

- switching to different cryptographic algorithms
  - aes-xts seems to be the fastest (at least on x86)
- experimenting with dm-crypt optional flags
  - "`same_cpu_crypt`" and "`submit_from_crypt_cpus`"
- trying filesystem-level encryption
  - much slower and potentially less secure

Despair

@ignatkn

Ask the community

“If the numbers disturb you, then this is from lack of understanding on your side. You are probably unaware that encryption is a heavy-weight operation...“

https://www.spinics.net/lists/dm-crypt/msg07516.html

But actually…

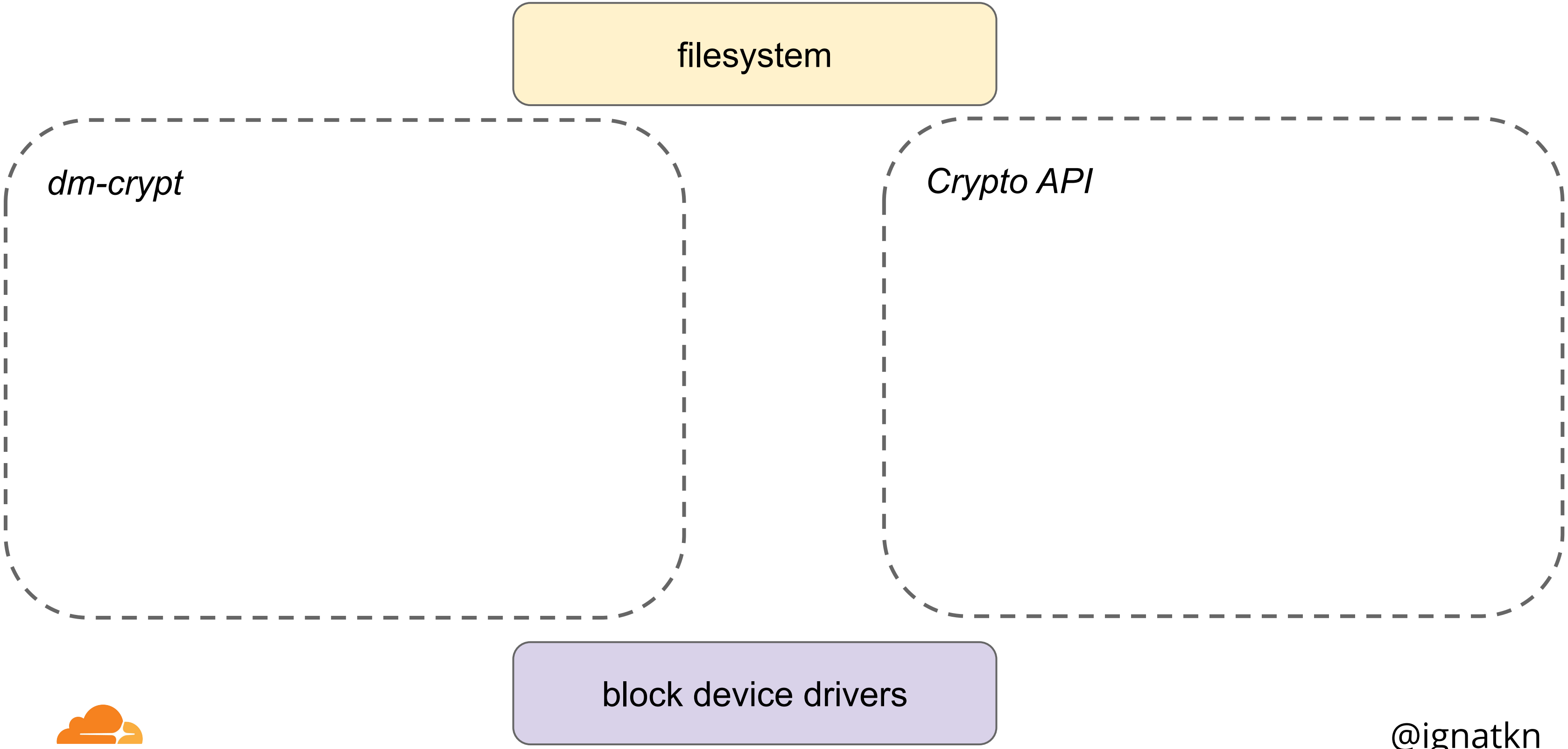
"Using TLS is very cheap, even at the scale of Cloudflare. Modern crypto is very fast, with AES-GCM and P256 being great examples."

https://blog.cloudflare.com/how-expensive-is-crypto-anyway/

dm-crypt: life of an encrypted BIO request

@ignatkn

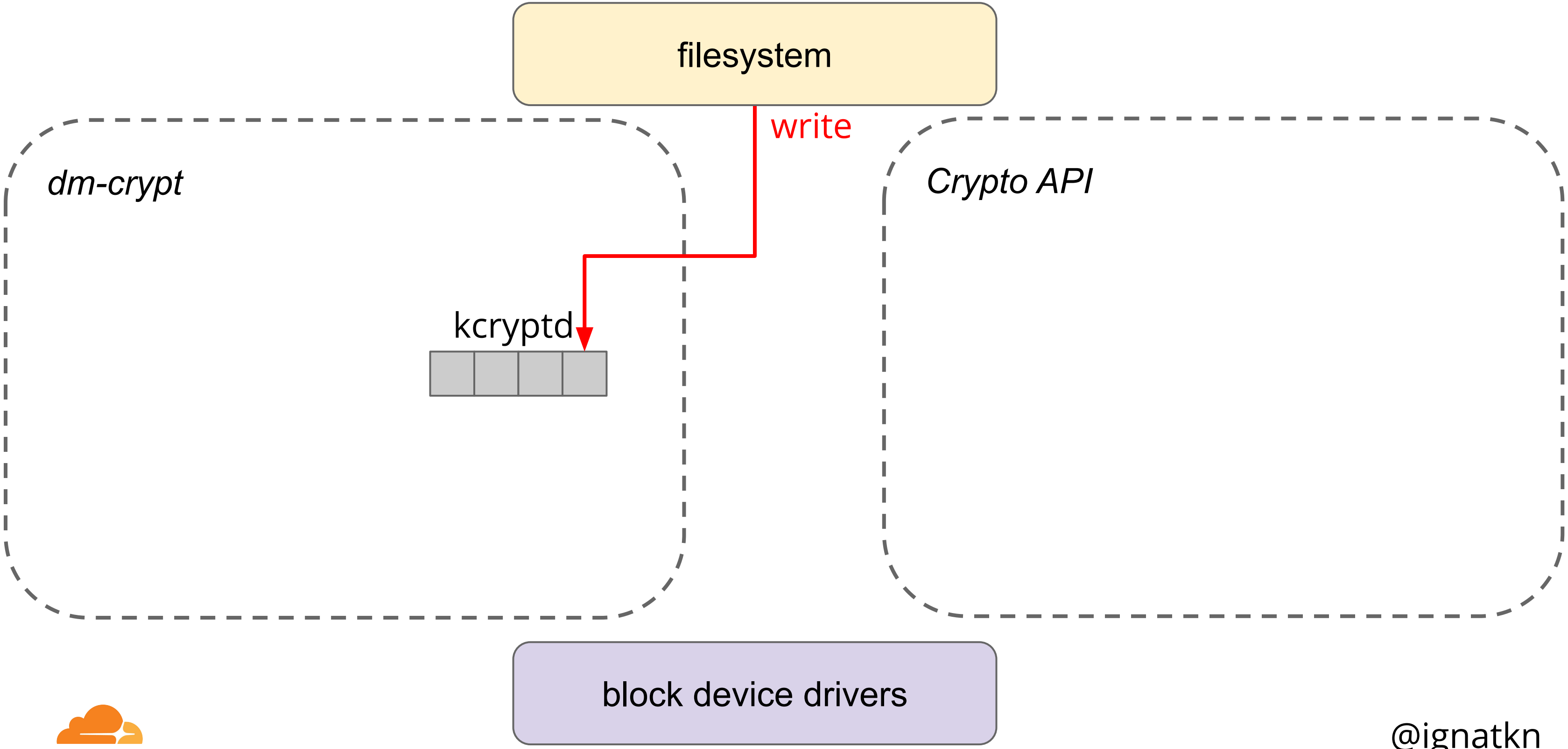
dm-crypt: life of an encrypted BIO request



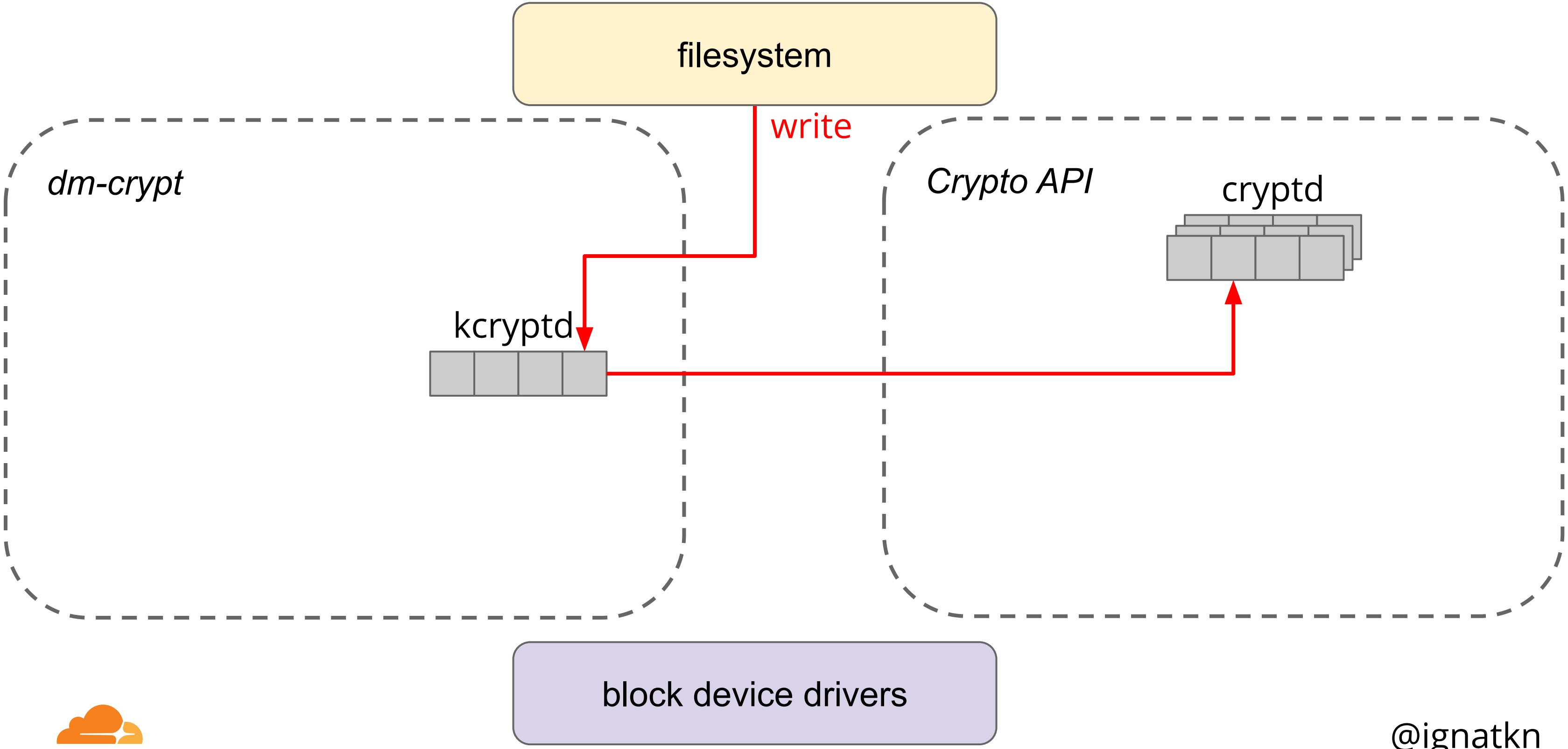
CLOUDFLARE

@ignatkn

dm-crypt: life of an encrypted BIO request



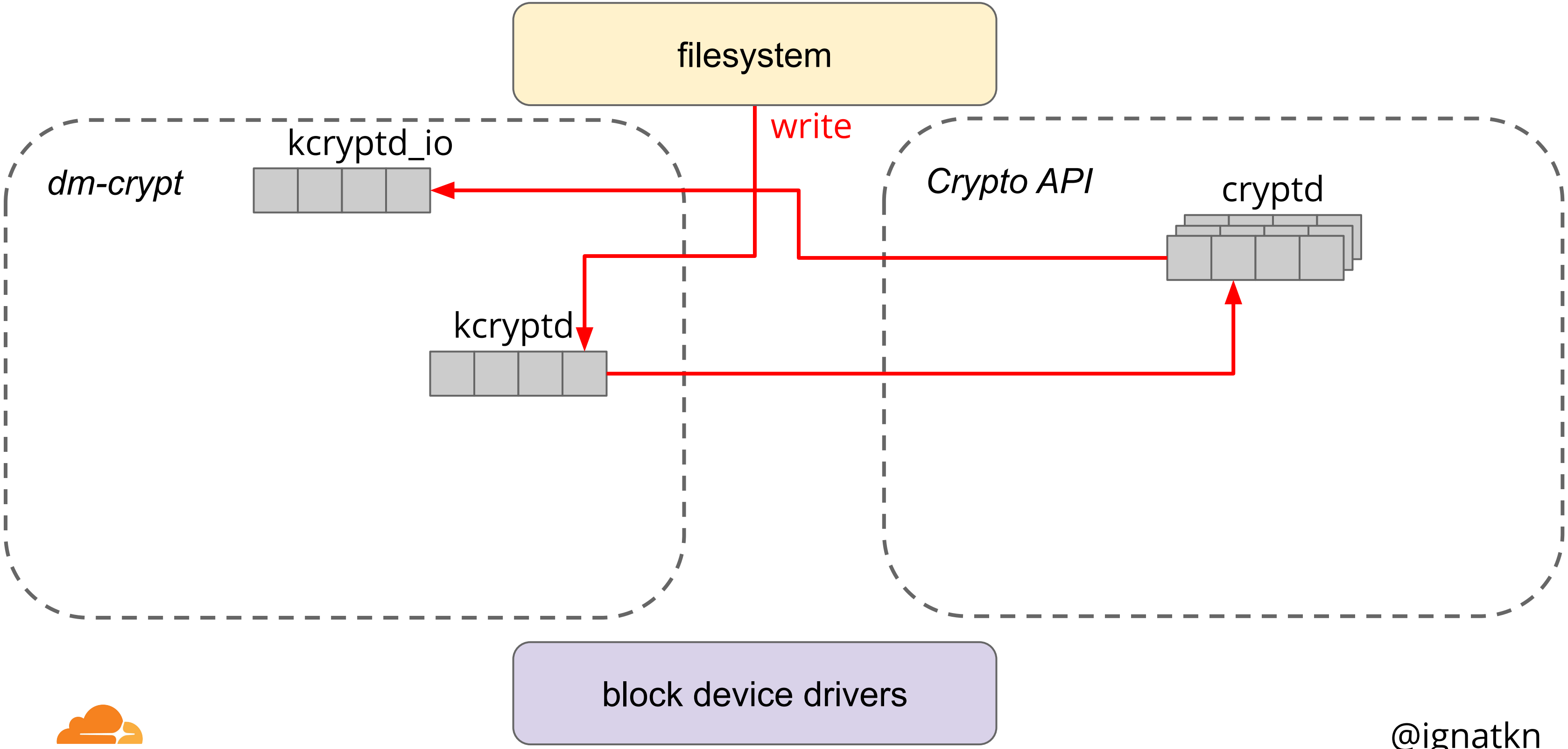
dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

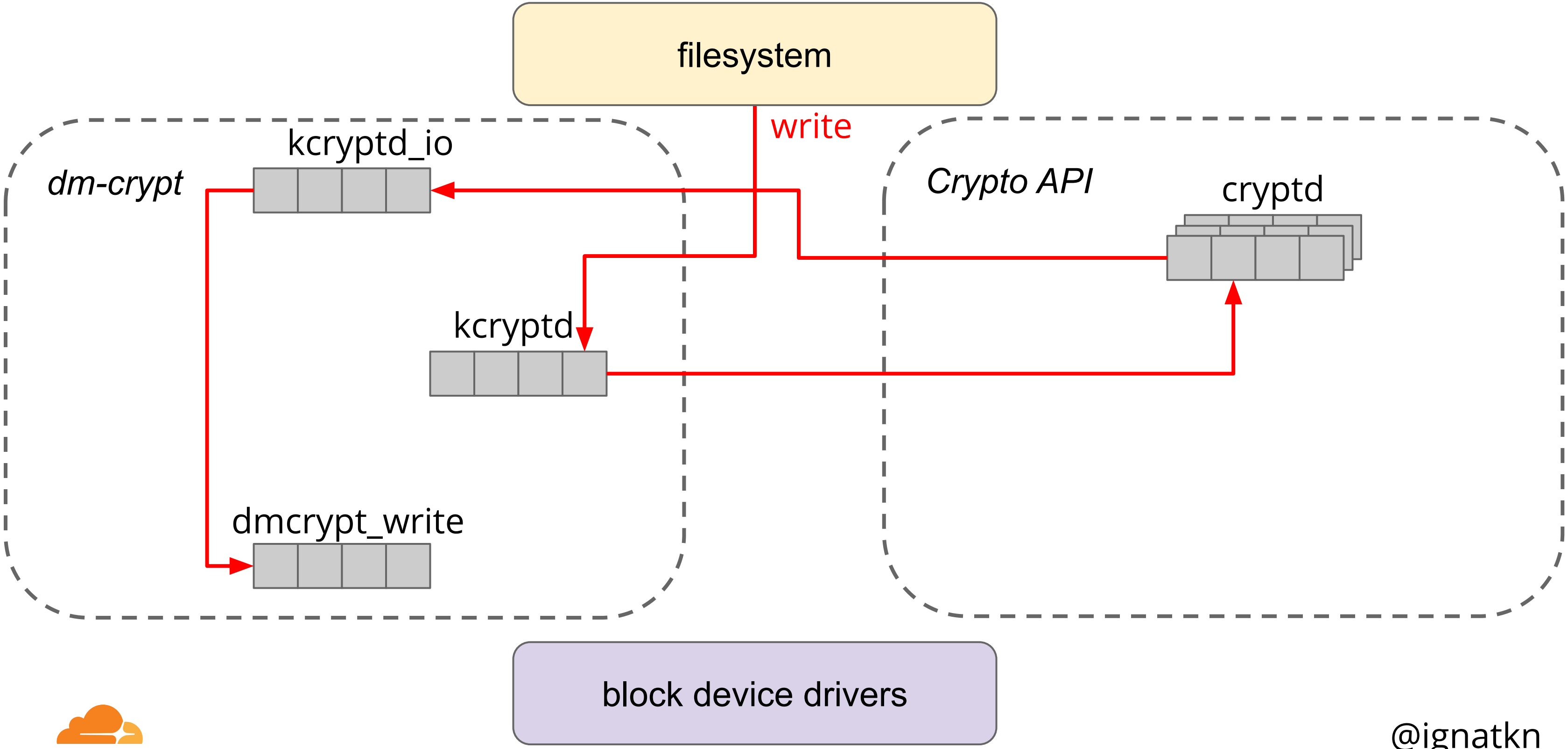
dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

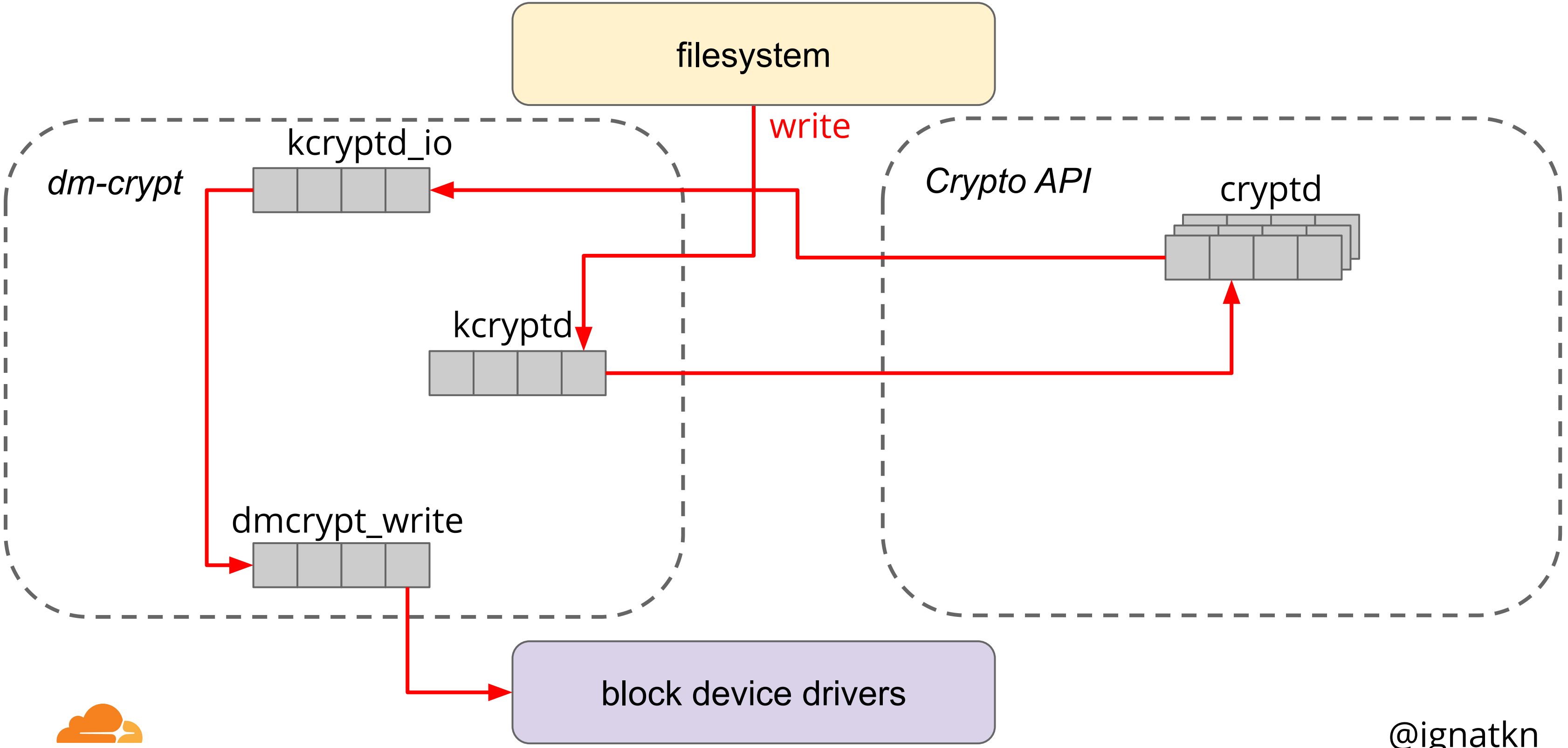
dm-crypt: life of an encrypted BIO request



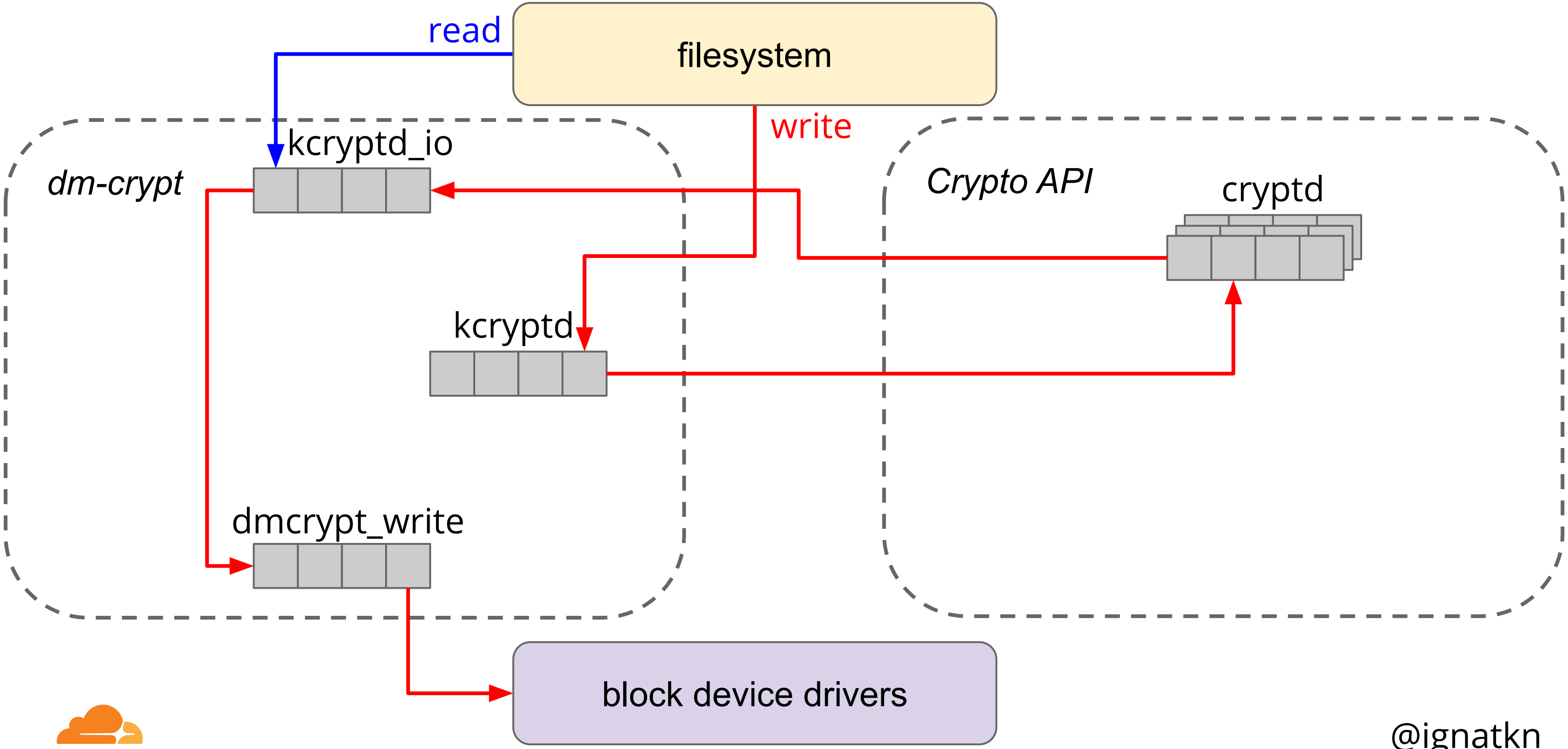
CLOUDFLARE

@ignatkn

dm-crypt: life of an encrypted BIO request



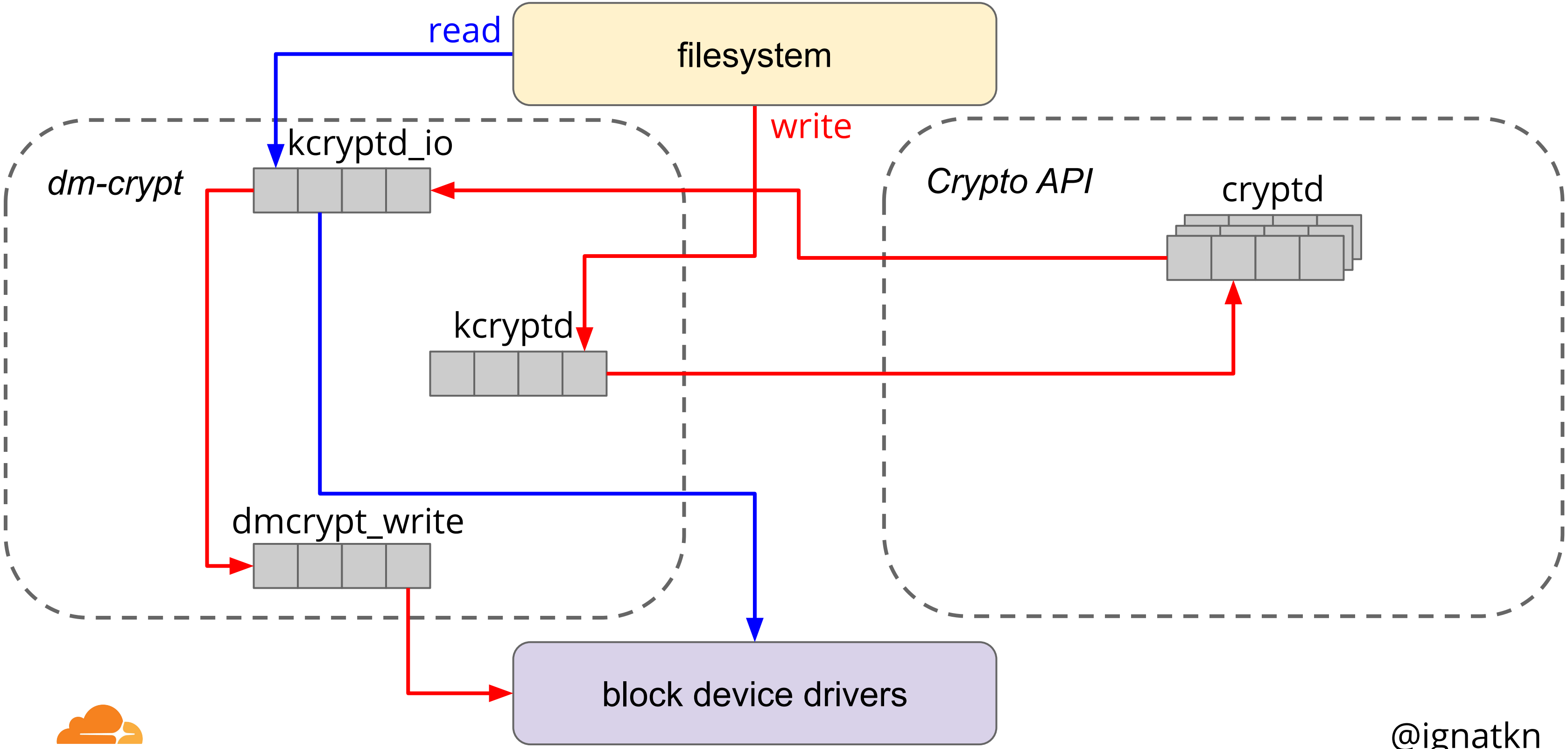
dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

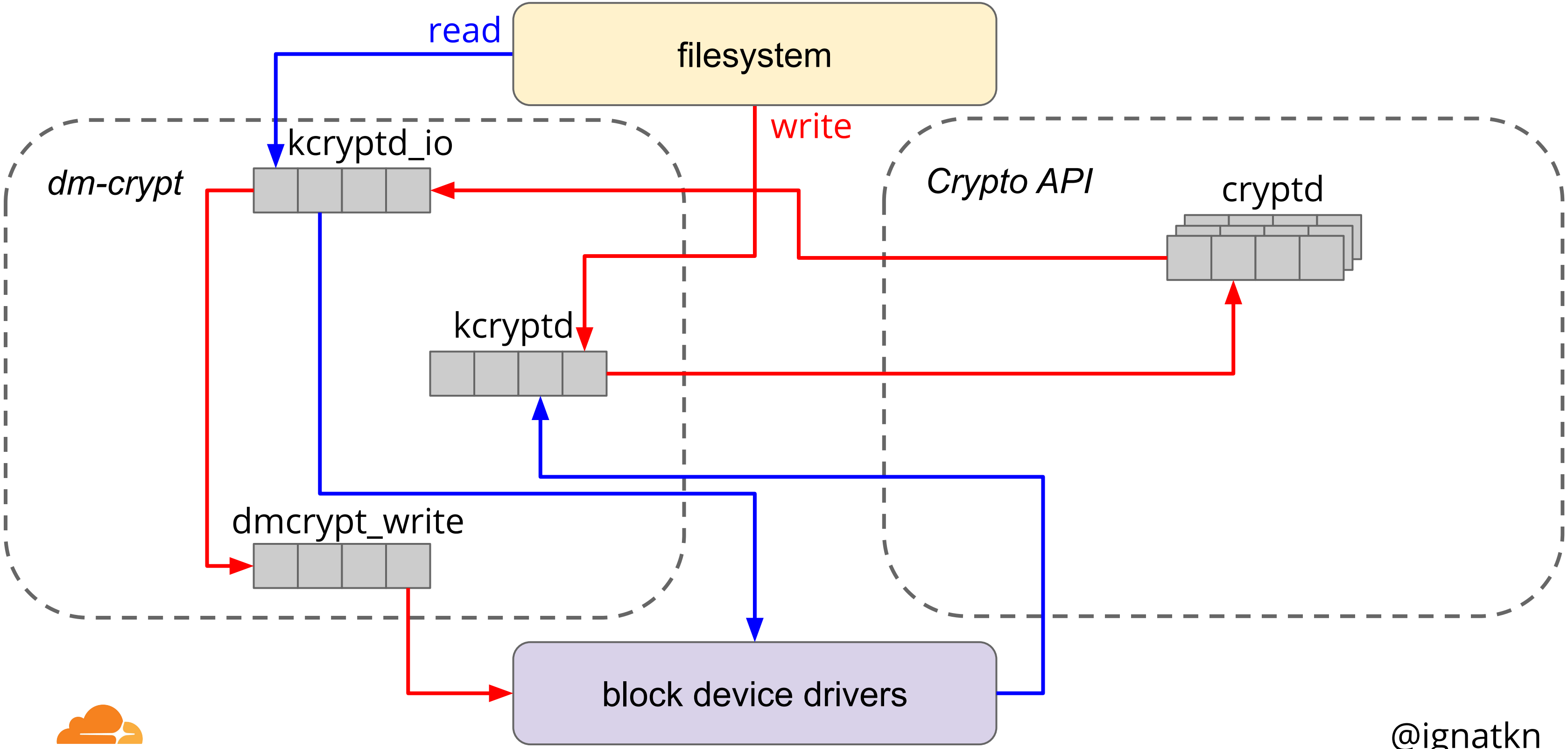
dm-crypt: life of an encrypted BIO request



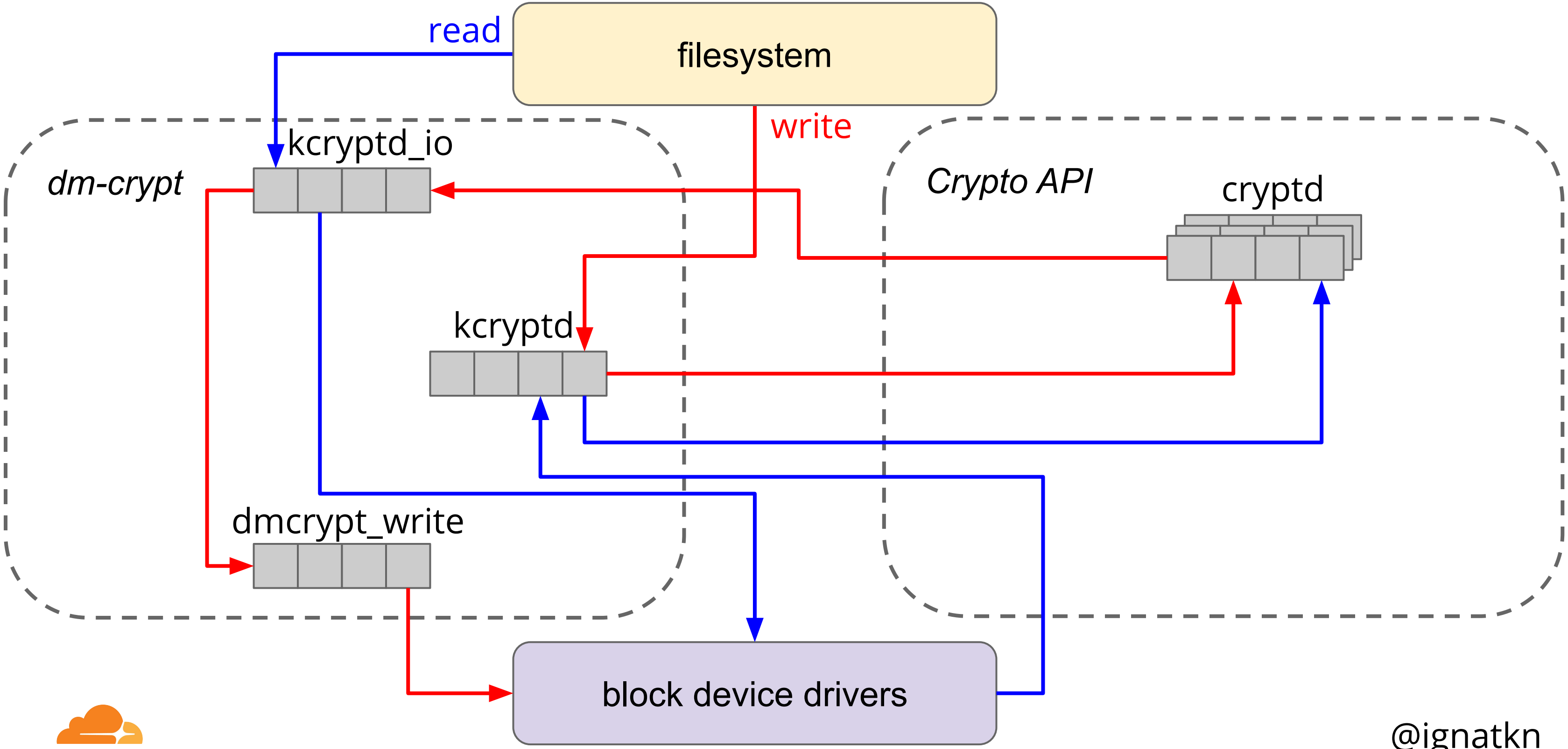
CLOUDFLARE

@ignatkn

dm-crypt: life of an encrypted BIO request



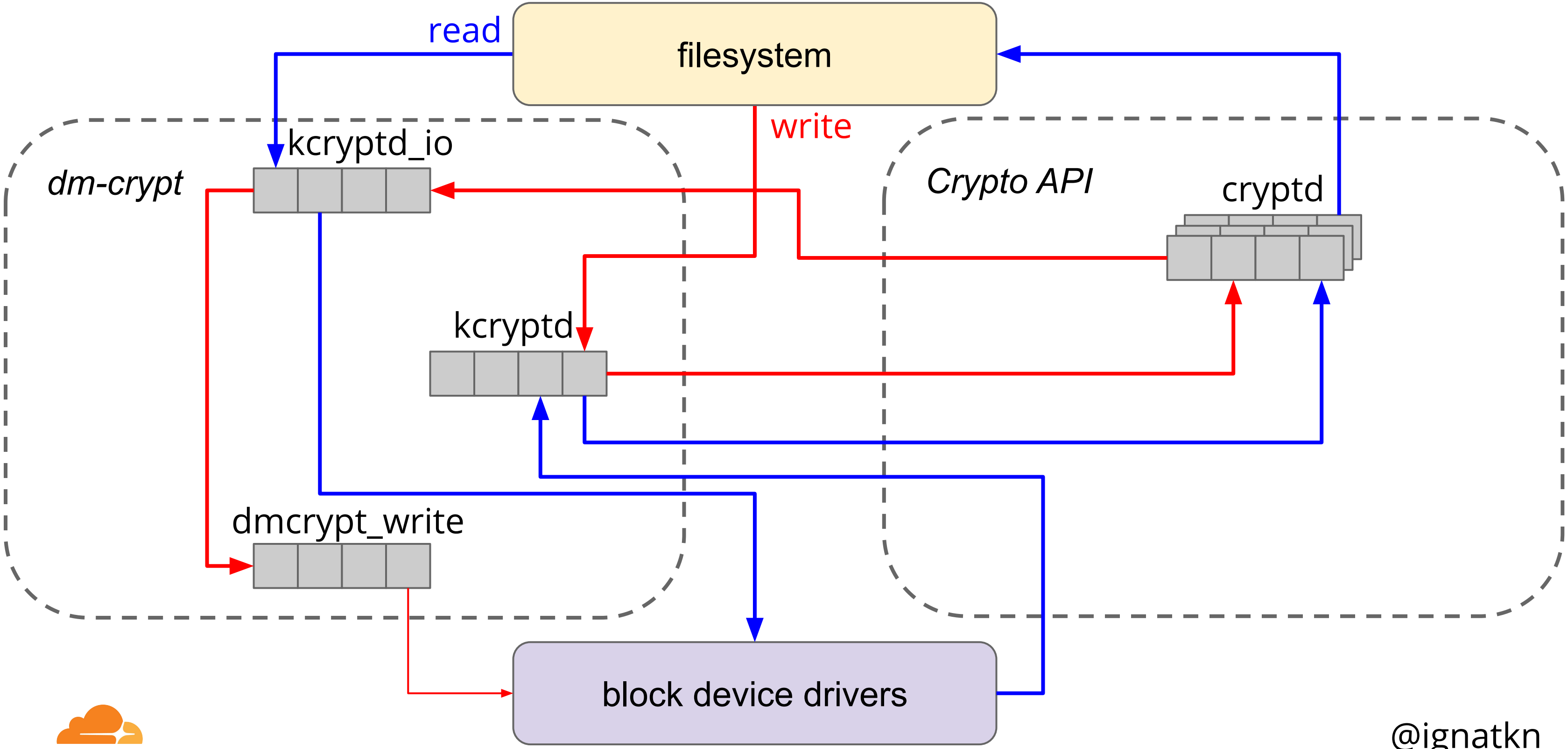
dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

dm-crypt: life of an encrypted BIO request



@ignatkn

queues vs latency

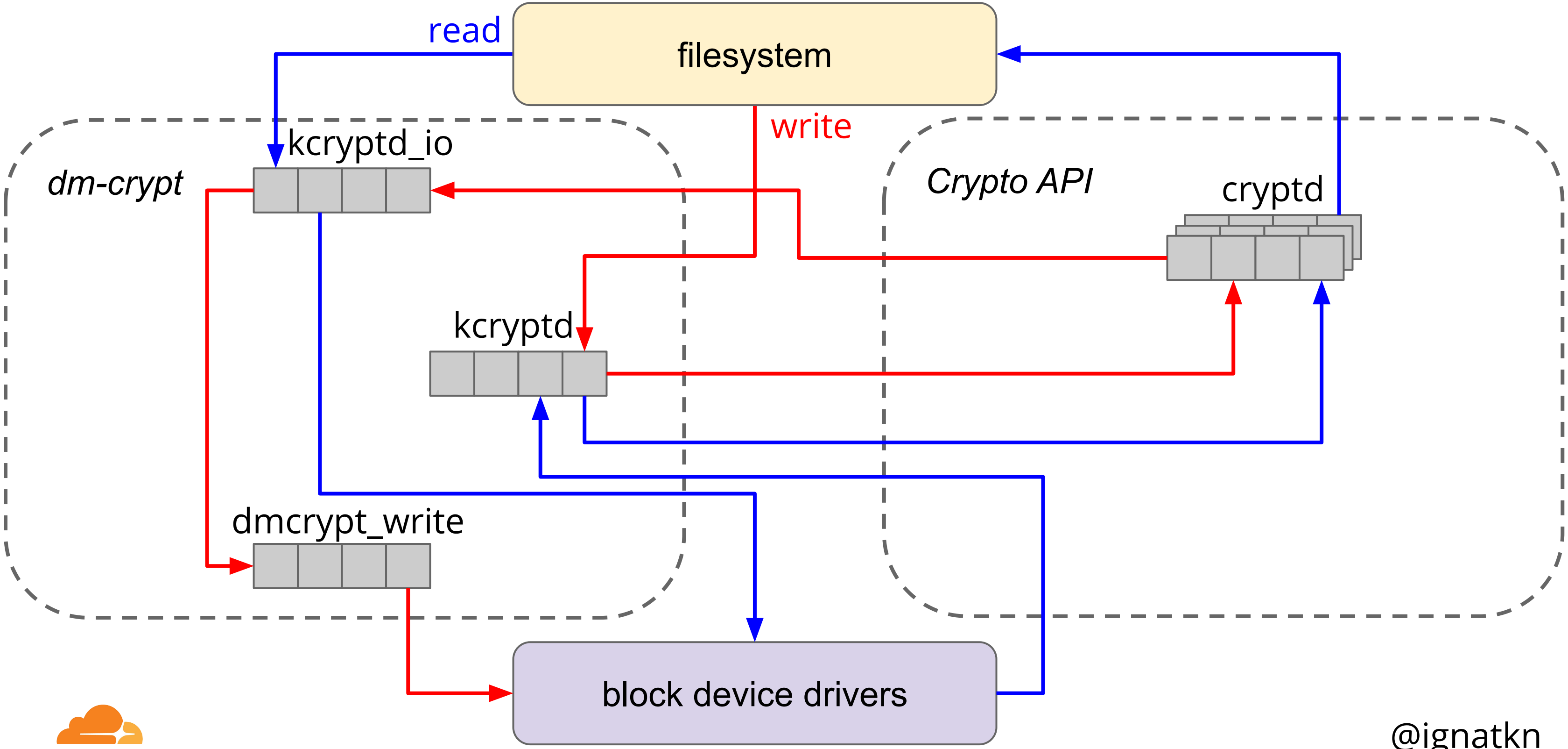
“A significant amount of tail latency is due to queueing effects”

https://www.usenix.org/conference/srecon19asia/presentation/plenz

CLOUDFLARE

@ignatkn

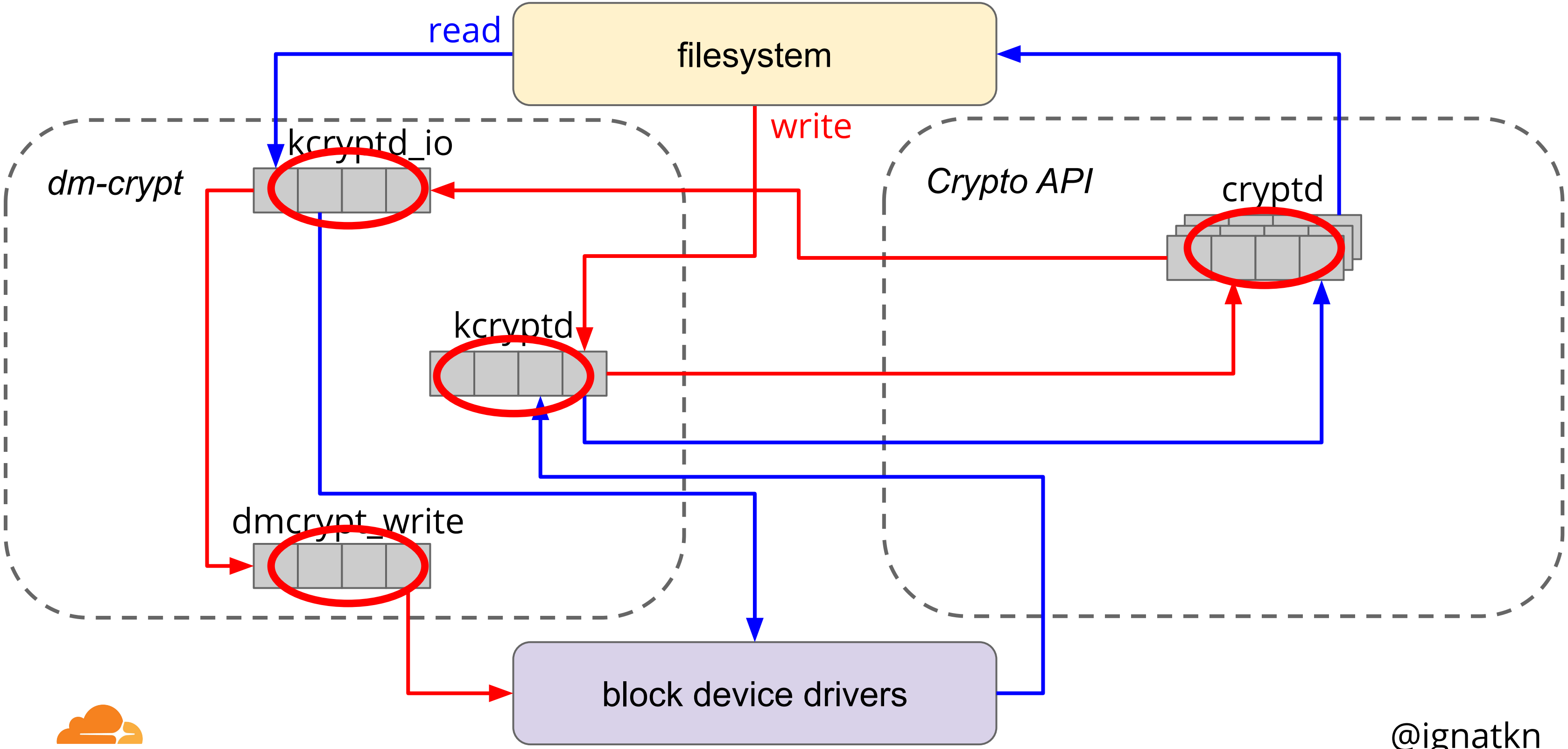
dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

dm-crypt: life of an encrypted BIO request



CLOUDFLARE

@ignatkn

dm-crypt: git archeology

- kcryptd was there from the beginning (2005)
  - only for reads: “it would be very unwise to do decryption in an interrupt context”

dm-crypt: git archeology

- kcryptd was there from the beginning (2005)
  - only for reads: “it would be very unwise to do decryption in an interrupt context”
- some queuing was added to reduce kernel stack usage (2006)

dm-crypt: git archeology

- kcryptd was there from the beginning (2005)
  - only for reads: “it would be very unwise to do decryption in an interrupt context”
- some queuing was added to reduce kernel stack usage (2006)
- offload writes to thread and IO sorting (2015)
  - for spinning disks, but “may improve SSDs”
  - mentions CFQ scheduler, which is deprecated

dm-crypt: git archeology

- kcryptd was there from the beginning (2005)
  - only for reads: “it would be very unwise to do decryption in an interrupt context”
- some queuing was added to reduce kernel stack usage (2006)
- offload writes to thread and IO sorting (2015)
  - for spinning disks, but “may improve SSDs”
  - mentions CFQ scheduler, which is deprecated
- commits to optionally revert some queuing
  - “`same_cpu_crypt`” and “`submit_from_crypt_cpus`” option flags

dm-crypt: things to reconsider

- most code was added with spinning disks in mind
  - disk IO latency >> scheduling latency

dm-crypt: things to reconsider

- most code was added with spinning disks in mind
  - disk IO latency >> scheduling latency
- sorting BIOs in dm-crypt probably violates “do one thing and do it well” Unix principle
  - the task for the IO scheduler

CLOUDFLARE

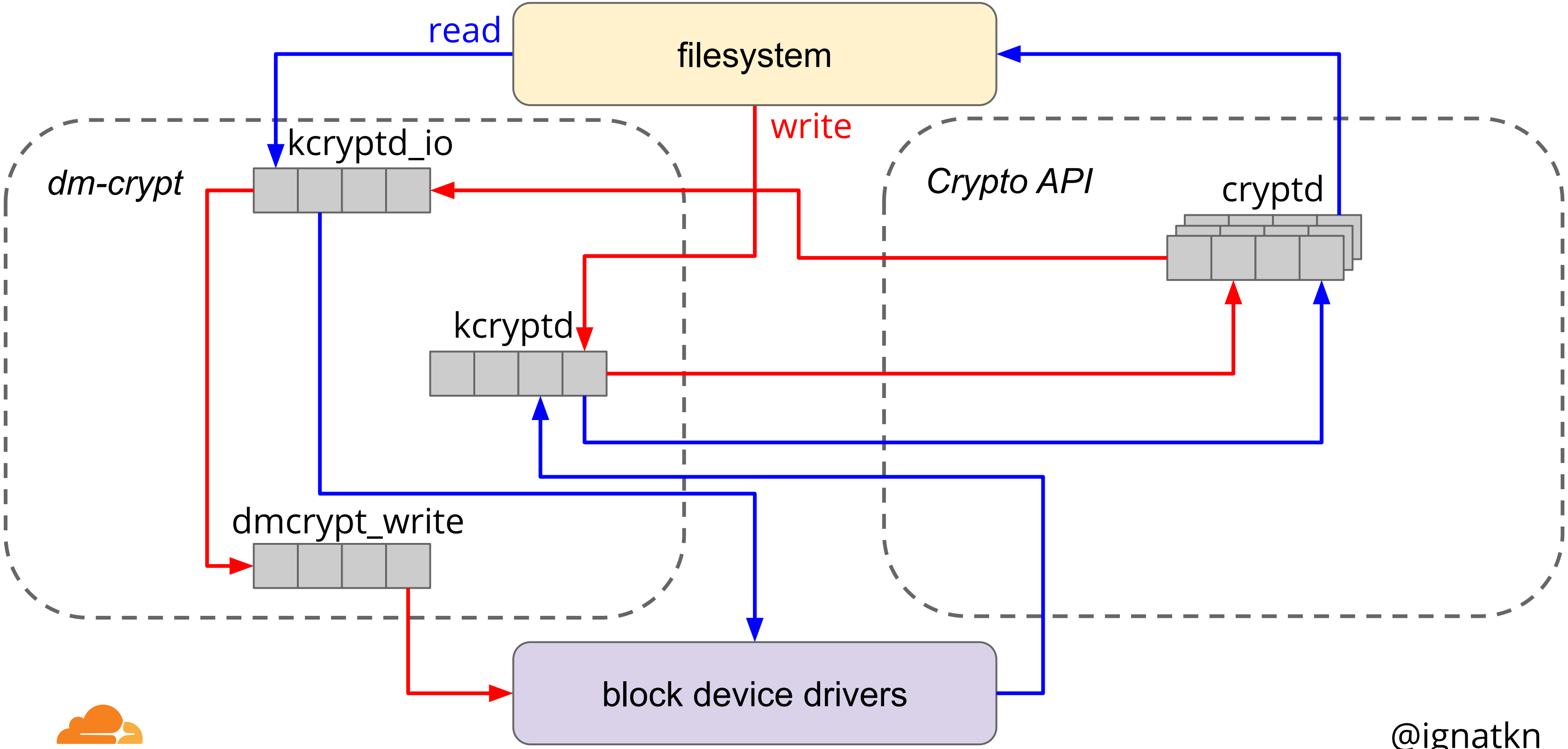
@ignatkn

dm-crypt: things to reconsider

- most code was added with spinning disks in mind
  - disk IO latency >> scheduling latency
- sorting BIOs in dm-crypt probably violates “do one thing and do it well” Unix principle
  - the task for the IO scheduler
- `kcryptd` may be redundant as modern Linux Crypto API is asynchronous by itself
  - remove offloading the offload

dm-crypt: cleanup

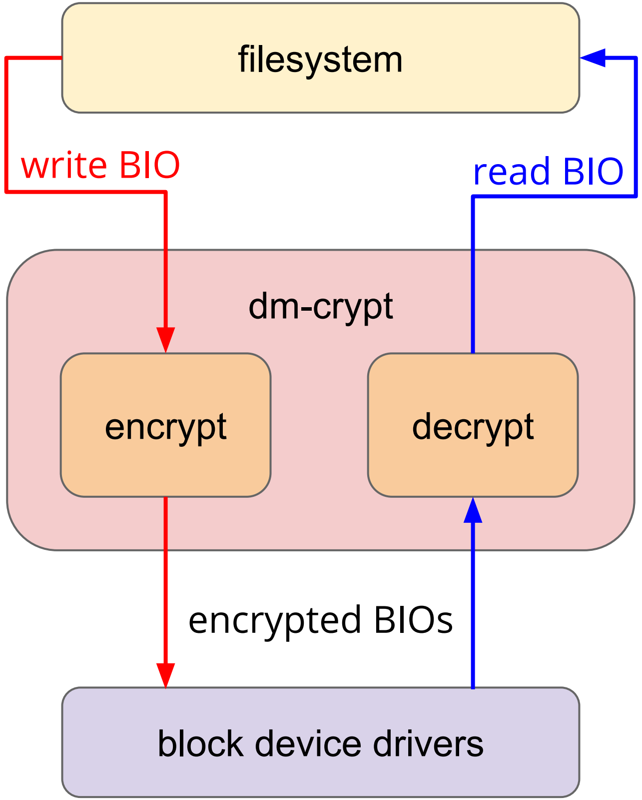
dm-crypt: life of an encrypted BIO request



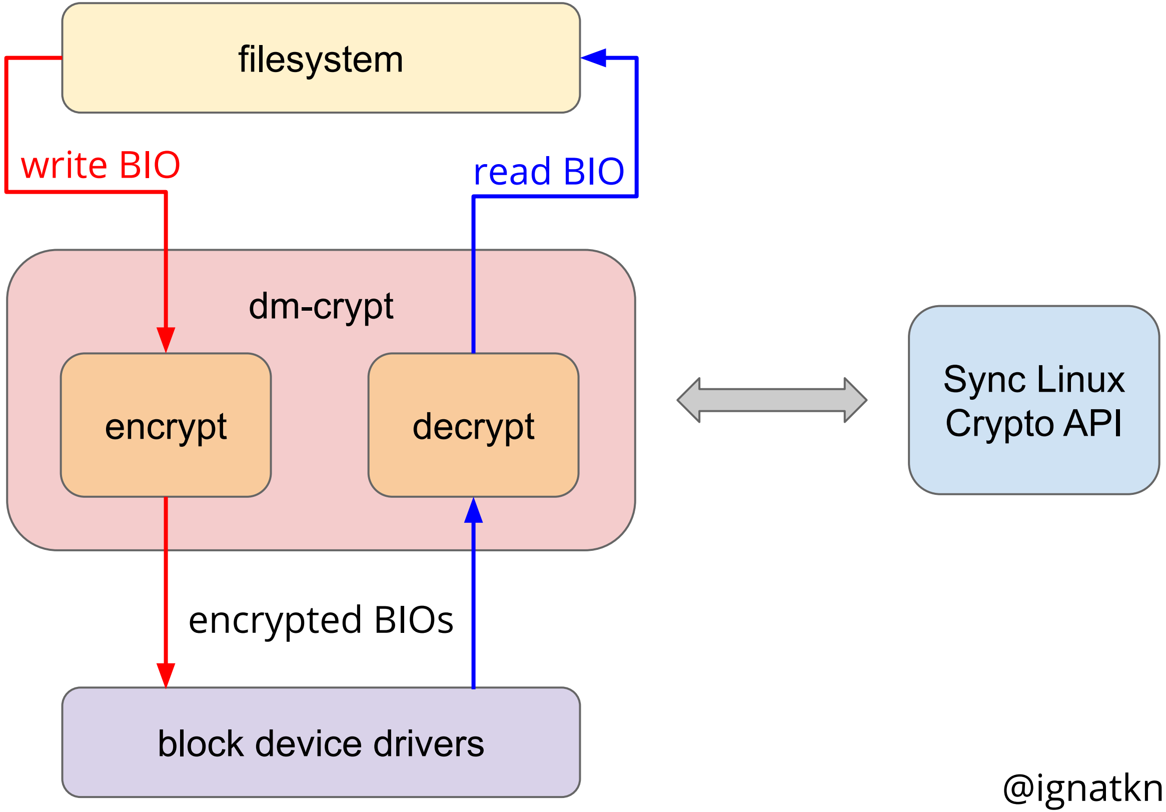
CLOUDFLARE

@ignatkn

dm-crypt (synchronous)



dm-crypt (synchronous)



@ignatkn

dm-crypt: removing queues

- dm-crypt module: a simple patch, which bypasses all queues/async threads based on a new runtime flag

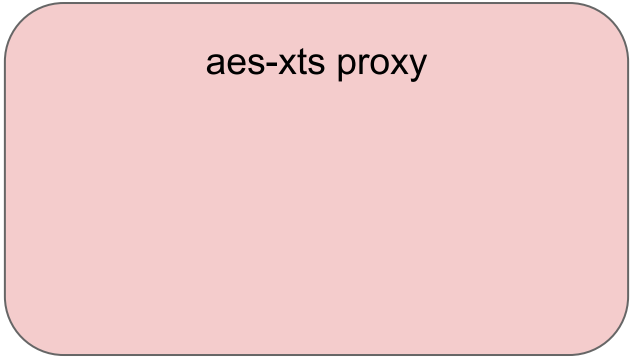
dm-crypt: removing queues

- dm-crypt module: a simple patch, which bypasses all queues/async threads based on a new runtime flag
- Linux Crypto API is a bit more complicated
  - by default specific implementation is selected dynamically based on priority
  - aes-ni synchronous implementation is marked as “internal”
  - aes-ni (FPU) is not usable in some interrupt contexts

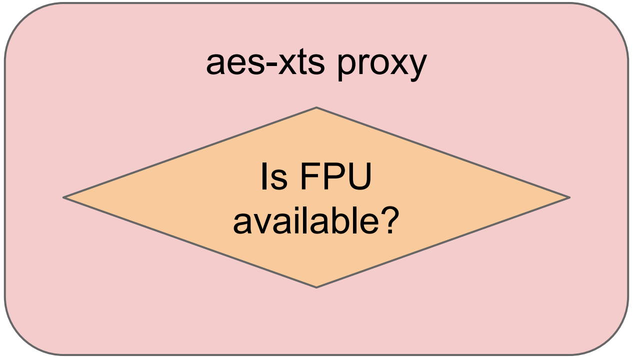
dm-crypt: removing queues

- dm-crypt module: a simple patch, which bypasses all queues/async threads based on a new runtime flag
- Linux Crypto API is a bit more complicated
    - by default specific implementation is selected dynamically based on priority
    - aes-ni synchronous implementation is marked as “internal”
    - aes-ni (FPU) is not usable in some interrupt contexts
- xtsproxy: a dedicated synchronous aes-xts module

xtsproxy crypto API module

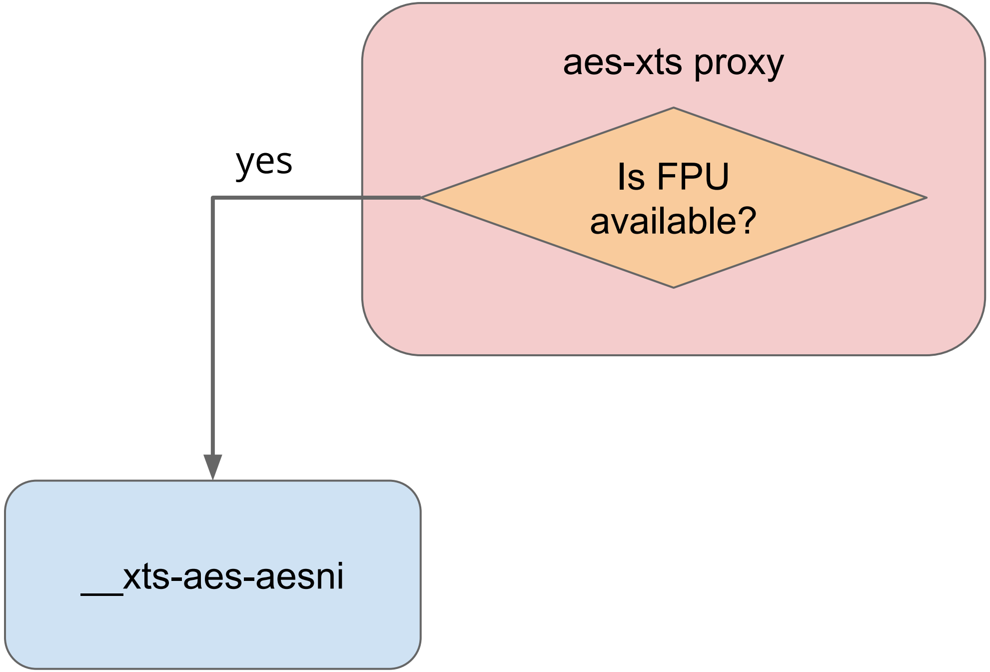


xtsproxy crypto API module

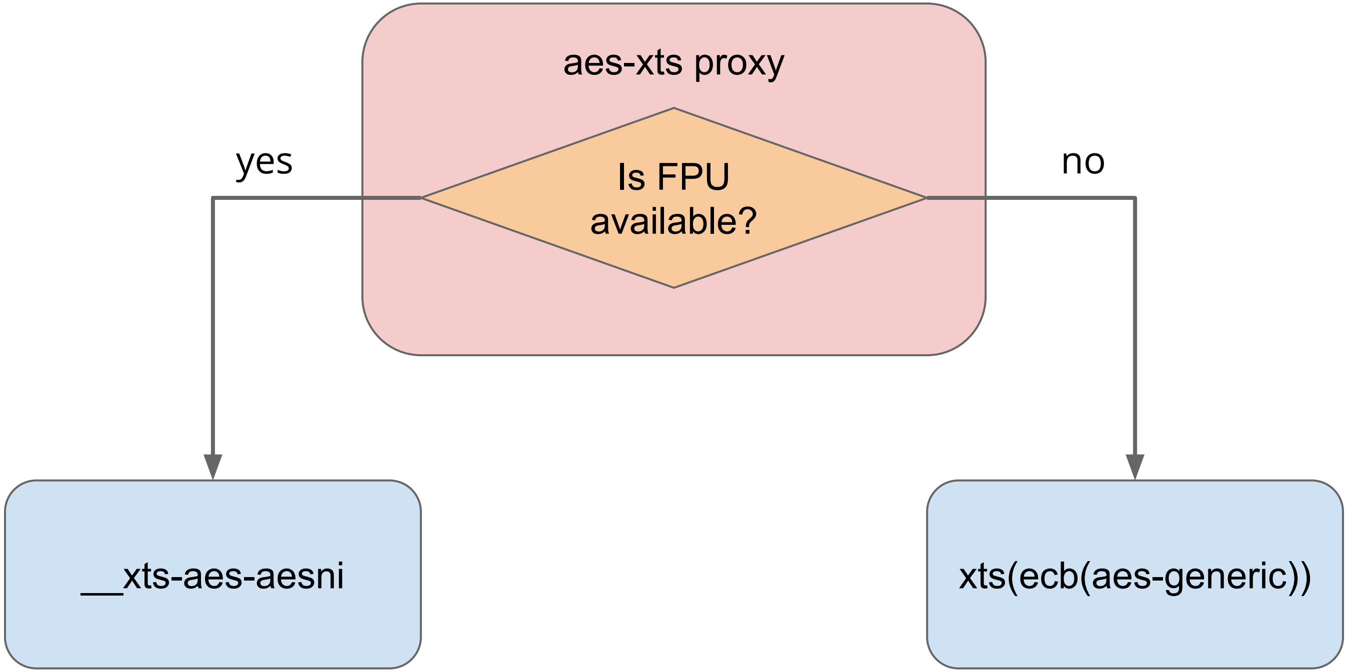


@ignatkn

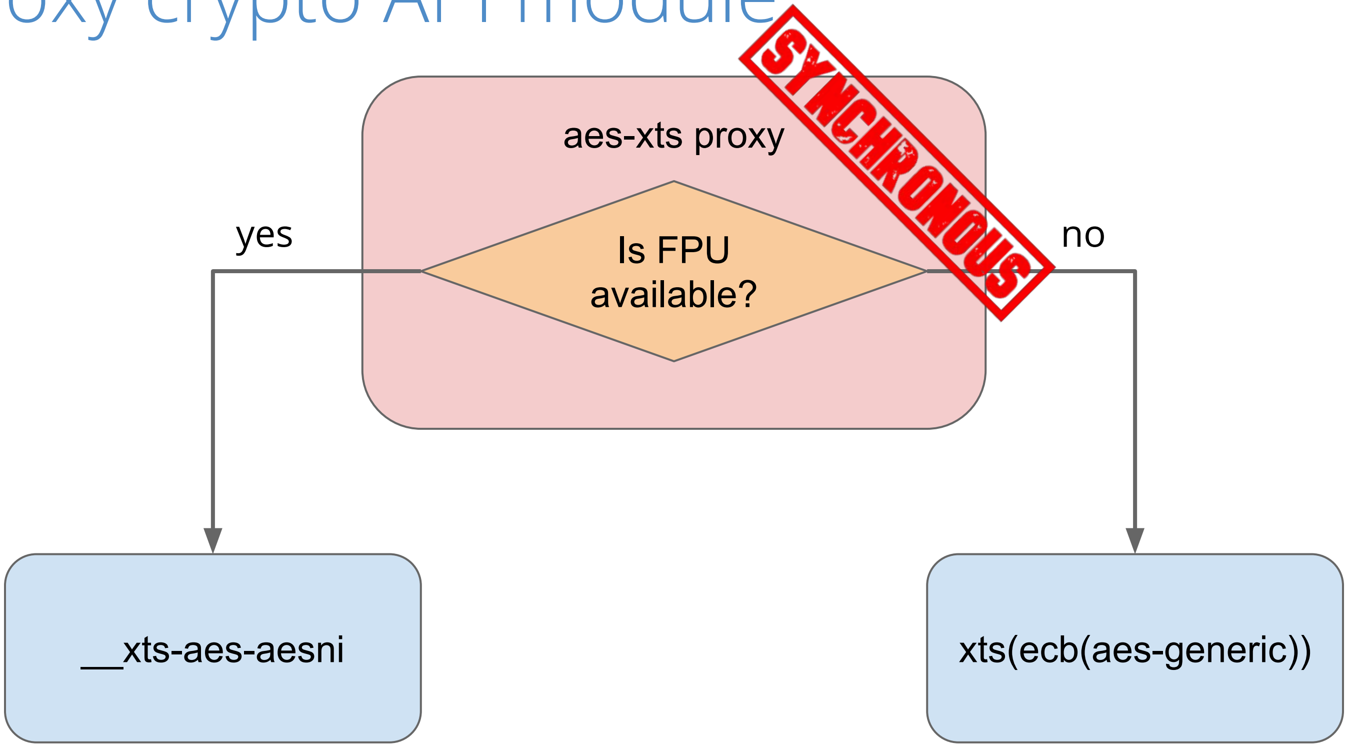
xtsproxy crypto API module



xtsproxy crypto API module



xtsproxy crypto API module



Test setup: sequential IO

```
$ sudo fio --filename=/dev/mapper/secure
--readwrite=readwrite --bs=4k rw.job
```

Test setup: sequential IO

```
$ sudo fio --filename=/dev/mapper/secure
--readwrite=readwrite --bs=4k rw.job
$ sudo modprobe xtsproxy
```

CLOUDFLARE

@ignatkn

Test setup: sequential IO

```
$ sudo fio --filename=/dev/mapper/secure
--readwrite=readwrite --bs=4k rw.job

$ sudo modprobe xtsproxy

$ sudo dmsetup table secure --showkeys | sed
's/aes-xts-plain64/capi:xts-aes-xtsproxy-plain64/' |
sed 's/$/ 1 force_inline/' | sudo dmsetup reload
secure
```

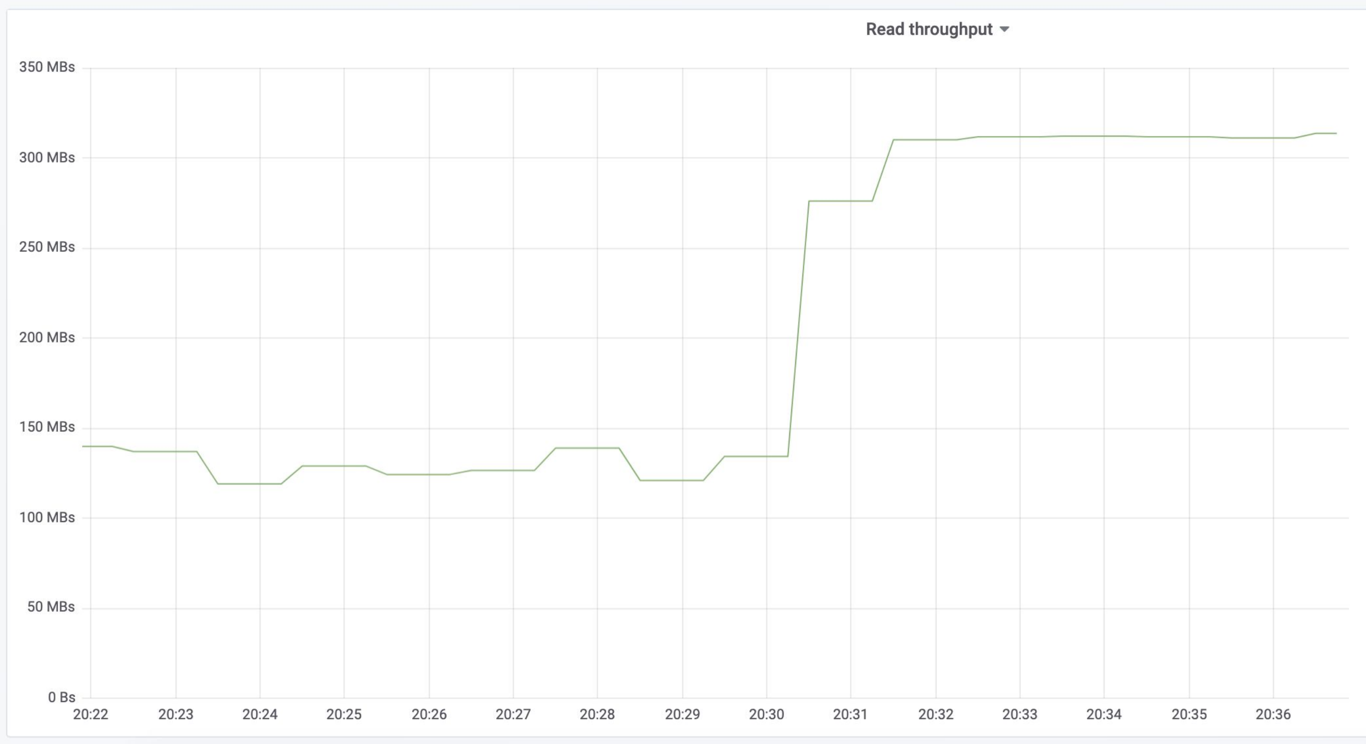
Test setup: sequential IO

```
$ sudo fio --filename=/dev/mapper/secure
--readwrite=readwrite --bs=4k rw.job

$ sudo modprobe xtsproxy

$ sudo dmsetup table secure --showkeys | sed
's/aes-xts-plain64/capi:xts-aes-xtsproxy-plain64/' |
sed 's/$/ 1 force_inline/' | sudo dmsetup reload
secure

$ sudo dmsetup suspend secure && sudo dmsetup resume
secure
```

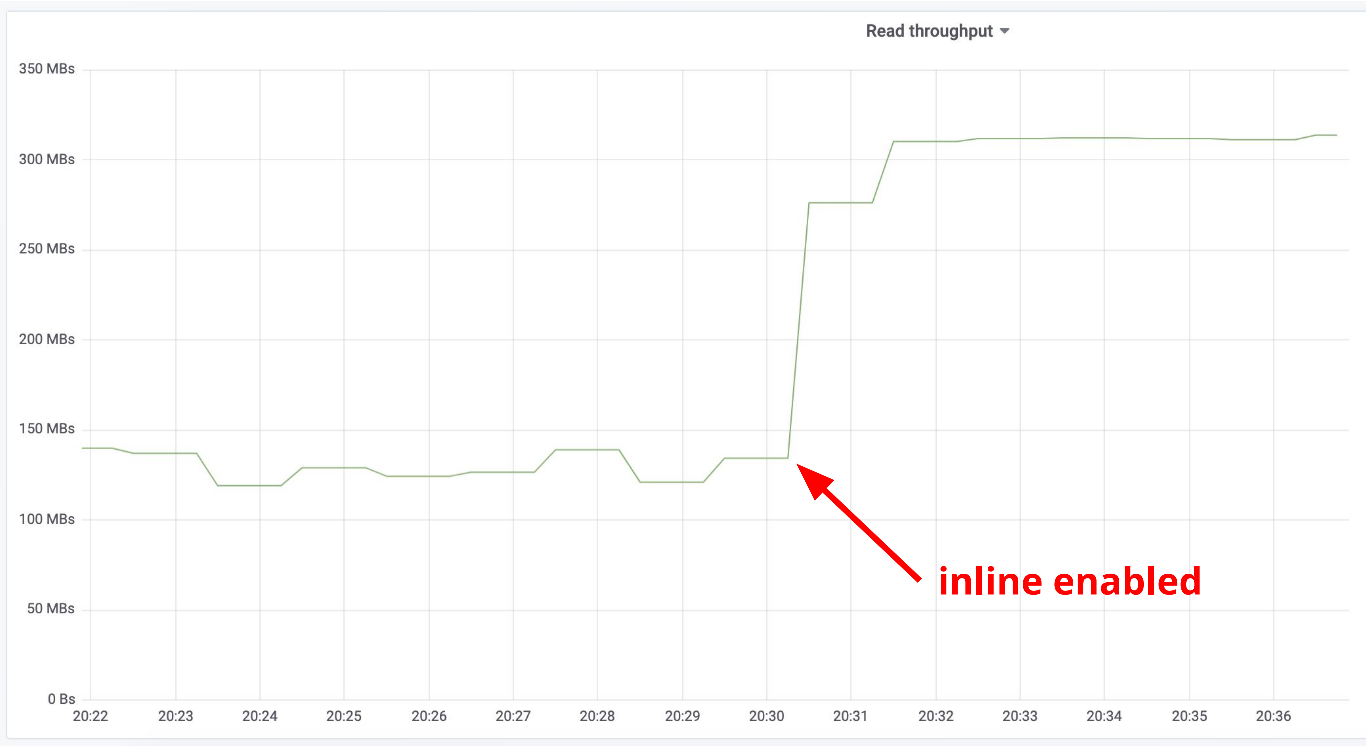
CLOUDFLARE

@ignatkn

ramdisk: read throughput



ramdisk: read throughput



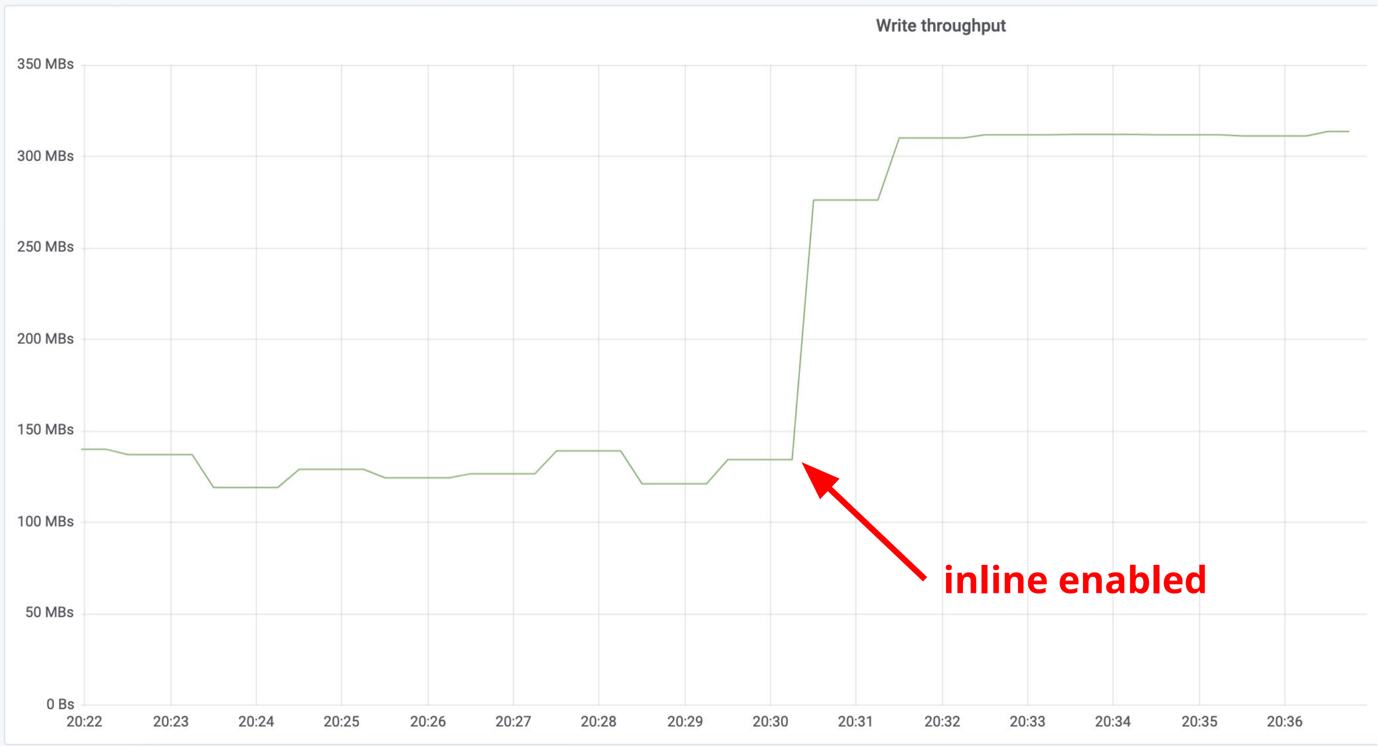
@ignatkn

ramdisk: write throughput

CLOUDFLARE

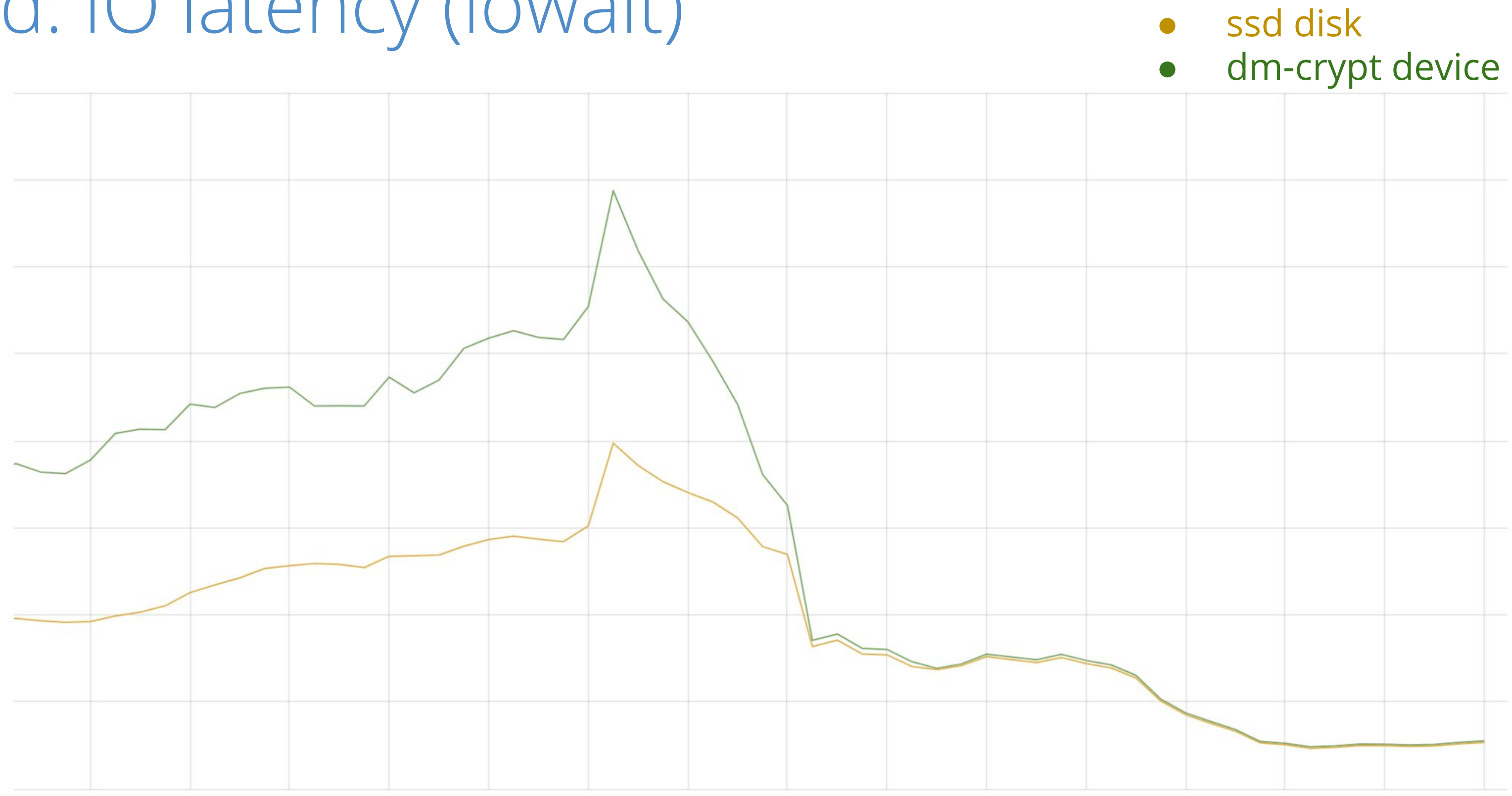
@ignatkn

ramdisk: write throughput



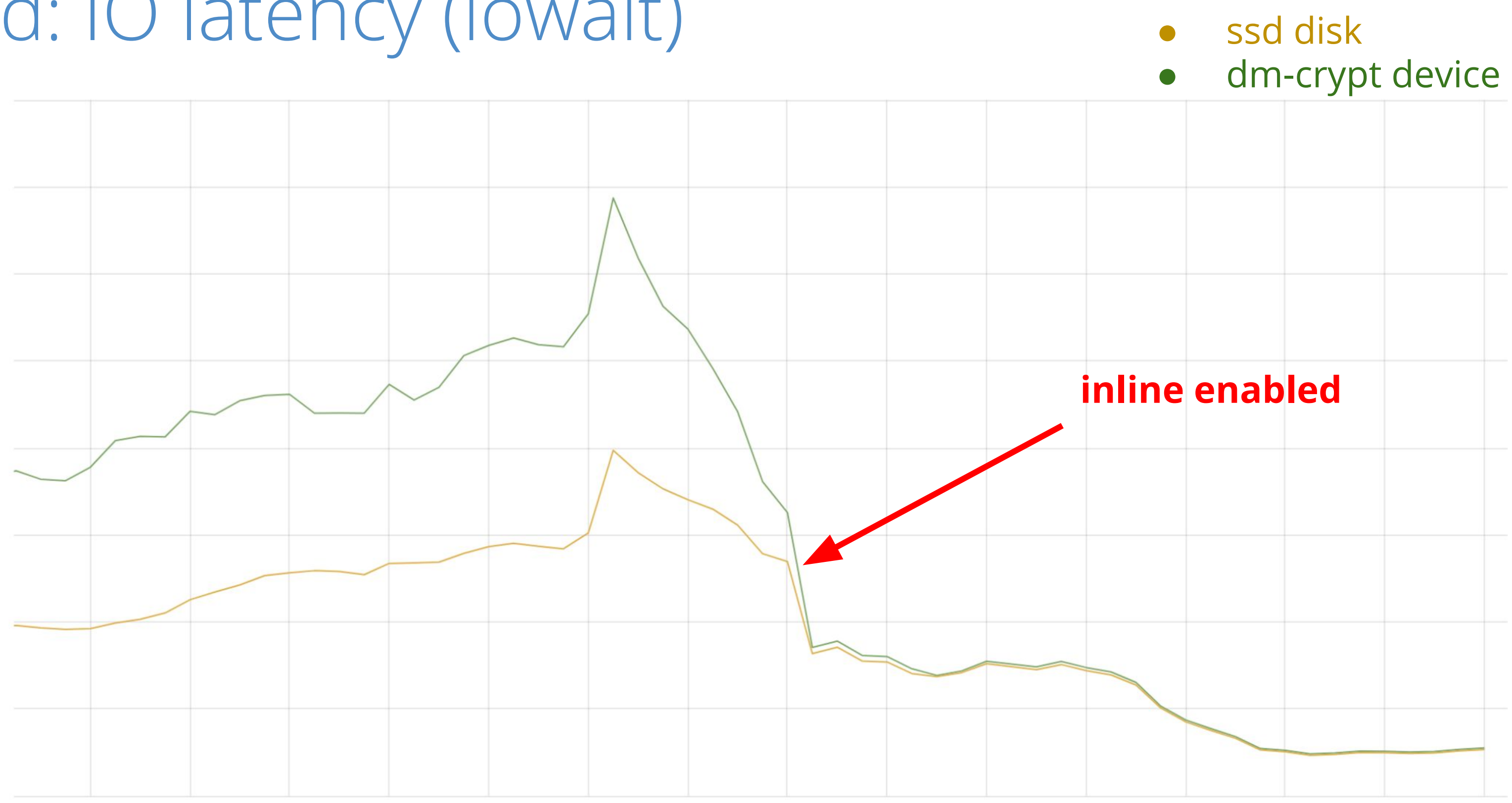
@ignatkn

ssd: IO latency (iowait)



@ignatkn

ssd: IO latency (iowait)



CLOUDFLARE

@ignatkn

Conclusions

Conclusions

- a simple patch which may improve dm-crypt performance by 200%-300%
  - fully compatible with stock Linux dm-crypt
  - can be enabled/disabled in runtime without service disruption

Conclusions

- a simple patch which may improve dm-crypt performance by 200%-300%
  - fully compatible with stock Linux dm-crypt
  - can be enabled/disabled in runtime without service disruption
- modern crypto is fast and cheap
  - performance degradation is likely elsewhere

CLOUDFLARE

@ignatkn

Conclusions

- a simple patch which may improve dm-crypt performance by 200%-300%
  - fully compatible with stock Linux dm-crypt
  - can be enabled/disabled in runtime without service disruption
- modern crypto is fast and cheap
  - performance degradation is likely elsewhere
- extra queuing may be harmful on modern low latency storage

Caveats and future work

- the patch improves performance on small block size/high IOPS workloads
  - >2MB block size shows worse performance

Caveats and future work

- the patch improves performance on small block size/high IOPS workloads
  - >2MB block size shows worse performance
- the whole setup assumes hardware-accelerated crypto
  - xtsproxy supports x86 only

Caveats and future work

- the patch improves performance on small block size/high IOPS workloads
  - >2MB block size shows worse performance
- the whole setup assumes hardware-accelerated crypto
  - xtsproxy supports x86 only
- your mileage may vary
  - always measure and compare before deployment
  - let us know the results

Links

- https://gitlab.com/cryptsetup/cryptsetup
- http://man7.org/linux/man-pages/man8/dmsetup.8.html
- https://github.com/cloudflare/linux

Questions?